Algoritmo de evacuaciones

Alejandro Morales Pérez

Grado en Ingeniería Informática

Facultad de Informática

UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo de Fin de Grado

Departamento de Ingeniería de Software

e Inteligencia Artificial

Septiembre 2017

Director:

Jorge J. Gómez Sanz

Autorizo a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionado expresamente a sus autores, tanto la propia memoria como el código, la documentación, contenidos audiovisuales y/o prototipo desarrollado.

Fdo. Alejandro Morales Pérez

Agradecimientos

Quiero agradecer el apoyo y la paciencia de mis padres y hermanos durante toda mi vida. Han sido, son y serán un pilar fundamental, sin este respaldo no hubiera alcanzado ninguna meta.

Agradecer también a amigos y compañeros que han sido parte de esta etapa, aportándome desde compañía y ayuda en momentos de trabajo hasta alegría y momentos de ocio.

A los profesores por su disponibilidad, paciencia y ayuda. Entre ellos a Jorge J. Goméz Sanz, sin su trabajo en este tema y experiencia, este trabajo no hubiera sido posible.

Resumen

Las personas, a lo largo de la historia, siempre se han encontrado ante situaciones de riesgo para sus vidas, como son los incendios. Con la finalidad de proteger sus vidas, existen estudios que intentan prevenir y entender el comportamiento de los seres humanos. Los compotarmientos estudiados van desde los que afectan solo a una persona, hasta los comportamientos de multitudes o pequeños grupos que surgen de manera espontánea, pero debido a la dificultad de recrear estas situaciones y a la complejidad del ser humano, no hay un consenso o una única corriente de pensamiento sobre ello. Las herramientas de simulación pueden ayudar a recrear estos desastres. Para el estudio de este campo y facilitar las evacuaciones ante estas situaciones, se tienen que tener en cuenta diversas variables que dan lugar a comportamientos como son, la experiencia, el conocimiento del lugar y del entorno, las relaciones sociales, entre otras.

Este es uno de los objetivos de este algoritmo o modelo de evacuaciones que sigue la teoría de la norma emergente de Killian y Turner, por la que individuos con intereses comunes tienen comportamientos similares dando lugar a normas emergentes y, implementa reacciones y factores estudiados en las dos guías de buenas prácticas del especialista Manuel Fidalgo. Este algoritmo permite la simulación de varios comportamientos y reacciones en desastres, sujetas a unas variables que representan características de las personas. El algoritmo implementado en Java y con simulaciones ejecutadas en la herramienta MASSIS, puede ser probado en diferentes tamaños de entornos y con varios tipos de incendios.

Este trabajo agradece el apoyo y mantenimiento de de la plataforma MASSIS desarrollada dentro del proyecto MOSI-AGIL (S2013/ICE-3019) con la financiación del gobierno de la Comunidad de Madrid y fondos FEDER. El proyecto es liderado por la Universidad Rey Juan Carlos y participan también la Universidad Politécnica y Universidad Complutense. El grupo GRASIA representa a la UCM en este consorcio. El objetivo del proyecto es crear tecnologías que puedan influenciar a los peatones en grandes instalaciones.

Palabras clave: evacuaciones, sistemas multi-agente, algoritmo, conducta colectiva, norma emergente, simulación, comportamiento, MASSIS.

Abstract

People throughout history have always been faced with situations of risk to their lives, such as fires. In order to protect their lives, there are studies that try to prevent and understand the behavior of humans. The studied behavior range from those that affect only one person, to the behavior of multitudes or small groups that arise spontaneously. Nevertheless to the difficulty of recreating these situations and the complexity of the human being, there is no consensus or a unique way of thinking about it. Simulation tools can help recreate these disasters. In order to study this field and facilitate evacuations in these situations, it is necessary to take into account diverse variables that trigger behaviors: the experience, the knowledge of the place and the environment, the social relations, among others.

This is one of the objectives of this algorithm or model of evacuations based the theory of the emerging norm of Killian and Turner, by which individuals with common interests have similar behaviors giving rise to emerging norms and, it implements reactions and factors studied in the two guides of good practices of the specialist Manuel Fidalgo. This algorithm allows the simulation of several behaviors and reactions in disasters, subject to variables that represent characteristics of the people. The algorithm was implemented in Java within the MASSIS tool. Using this tool, different fire evacuation scenarios were tested.

This work acknowledges the support of the MASSIS framework developed under the MOSI-AGIL (S2013/ICE-3019) project funded by the Government of the Region of Madrid, and European Structural Funds (FEDER). The project is leaded by the Universidad Rey Juan Carlos, and participated by Universidad Politécnica de Madrid and Universidad Complutense de Madrid. GRASIA research group represents the UCM in this consortium. The goal of the project is to create technologies that can influence pedestrians in their walking habits within a large facility.

Key words: evacuations, multi-agent systems, algorithm, collective behavior, emergent norm, simulation, behavior, MASSIS.

Índice

Índice de figuras

Índice de tablas

Índice de abreviaturas

MAB	Model Agent-Based
IDE	Integrated Development Environment/Entorno de Desarrollo Integrado
SMA	Sistemas Multiagente
IAD	Inteligencia Artificial Distribuida
IA	Inteligencia Artificial
GRASIA	Research Group on Agent-Based, Social and Interdisciplinary Applications
UCM	Universidad Complutense de Madrid
CCL	Center for Connected Learning and Computer-Based Modeling
CORMAS	COmmon Resources Multi-Agent System

BOID	Bird-OID Object
MANTA	Model of ANThill Activity
EMF	EthoModeling Framework
MASON	Multi-Agent Simulator Of Neighborhoods
MASSIS	MultiAgent System Simulation of Indoor Scenarios
JSON	JavaScript Object Notation
E/S	Periférico de entrada/salida
JADE	Java Agent Development Framework
JAFMAS	Java Framework for Multi-agent Systems
MADkit	Multi-Agent Development Kit

1. Introducción

1.1. Motivación.

El ser humano desde la Época Clásica trata de entender y explicar patrones que se repiten entre personas que forman una sociedad. Estos estudios, bajo el término general de conducta colectiva, han creado disciplinas en la psicología, la psicología social y la sociología. A partir de principios del siglo XX, se pueden destacar trabajos como son los del psicólogo y sociólogo estadounidense George Herbert Mead [1], que resalta la importancia de la aceptación social de la opinión de cada inviduo. Más adelante, el psicólogo británico William McDougall [2], aseguraba que todos los miembros de una sociedad hacen uso de sus funciones fisiológicas para adaptarse al medio. Cabe destacar al padre del psicoanálisis, el austríaco Sigmund Freud [3], que quería conocer la necesidad de los deseos sexuales del hombre. Entre finales del siglo XX y principios del siglo XXI, destacan varios estudios sobre simulaciones de evacuaciones. Guylene Proulx [8] resalta la importancia de una buena señalización de las salidas en el edificio, un sistema de comunicación eficiente y el grado de experiencia o entrenamiento del personal. Steve Gwynne et al. [9][10] afirman que los vínculos sociales antes y durante la evacuación influyen en las decisiones y acciones adoptadas en respuesta al desastre.

En la actualidad, se encuentran dos grandes modelos para entender la conducta colectiva [4]. Por una parte, el modelo que proviene de la psicología de las masas, cree que la conducta colectiva es anómala, irracional, emocional, asocial y patológica, por lo que necesita de estrategias de intervención externas proclives al control y la prevención. Por otra parte, el modelo que proviene de algunas teorías de la sociología y de la sociología del desastre posterior, defiende la conducta colectiva como una forma de conducta social que se caracteriza por la normatividad, la sociabilidad, la racionalidad, la cooperación y los comportamientos prosociales. Las estrategias de intervención recomendadas para este modelo se fundamentan en una organización democrática, participativa y no catastrófista. De este modelo último existen un mayor número de estudios empíricos. La teoría de la

norma emergente de Turner y Kilian [18] se apoya en modelo, en ella se defiende que pueden emerger conductas en multitudes si en los individuos que la forma existen intereses comunes, dado que las multitudes no son irracionales.

Para obtener un estudio estricto y válido sobre conducta colectiva es necesario la combinación de diferentes alcances para los análisis de comportamientos intergrupales, comportamientos intragrupales, comportamientos interindividuales e individuales. Así como también, ajustar el tipo de fenómeno (desastres naturales, linchamientos en movimientos sociales, incendios provocados por el ser humano, etc), tipo de multitud (conferencias, centro de estudios o trabajo, centro de ocio, etc) y tipo de situación de la multitud (peatones en tránsito, multitud congregada de forma espontánea o intencionada, evacuación de grupos). Cabe destacar como ejemplo de estudios que engloban estos alcances comentados anteriormente, los dos que ha realizado Manuel Fidalgo sobre el comportamiento humano en emergencias, en el primero de ellos analiza la conducta individual y las características que influyen en ella como son percepción del riesgo, reacción, medidas preventivas, entre otras [43]. En el segundo estudio, Fidalgo tiene como objetivo el estudio de la conducta colectiva, para el autor está influenciada por el tipo de multitud, lugar, contagio mental, entre otras características [44]. Estos dos trabajos son la base para el desarrollo del algoritmo o modelo de evacuación de este proyecto.

En la sociedad contemporánea, debido a las herramientas tecnológicas con las que se cuentan, es posible simular y precisar algo más en los resultados de estos estudios. Entre estas herramientas se encuentran dos tipos, los sistemas multiagente (SMA) y las aplicaciones que se rigen por el modelo basado en agentes (MAB).

Los SMA son sistemas informáticos formados por un grupo de agentes que interactúan entre sí utilizando protocolos y lenguajes de comunicación de alto nivel, para resolver probemas que pueden estar fuera de las capacidades o del concimiento de cada agente [11]. Algunas herramientas de tipo SMA son JADE, JAFMAS y MADkit [12].

MAB es un tipo de modelo computacional que permite simular acciones e interacciones de individuos autónomos dentro de un entorno, y permite determinar qué efectos producen en el conjunto del sistema [5][6]. Este modelo está basado en áreas como teoría de juegos, sistemas complejos, emergencia, sociología computacional, sistemas multi-agente, y programación evolutiva. Los agentes MAB pueden desarrollar aprendizaje, adaptación y reproducción. Algunos ejemplos de aplicaciones MAB son AnyLogic, GAMA, VisualBots y MASSIS.

MASSIS es una aplicación de software, creada por Rafael Pax, que facilita la

simulación de escenarios con múltiples entidades en entornos interiores, dentro de edificio. Esta aplicación proporciona soporte para diseñar el entorno y elegir el comportamiento de los elementos y entidades [42]. MASSIS (MultiAgent System Simulation of Indoor Scenarios) permite recrear una gran variedad de comportamientos, desde un sensor hasta las decisiones de una persona. Ha sido diseñado para mantener esta flexibilidad sin dificultar el rendimientos, siendo capaz de simular miles de entidades, cada una con un comportamiento específico. Esto último refuerza la idea de crear comportamientos tanto individuales como colectivos para un número amplio de agentes bajo una carecterísticas determinadas en la herramienta, objetivo que se busca en este trabajo.

Entre los trabajos de simulación basados en agentes, existen los que recrean bandadas de pájaros en vuelo como la herramienta Boids, creada por Craig Reynolds [27]. Además, otros trabajos producen comportamientos o dinámicas sociales, como son el modelo de segregación de Thomas Schelling, también llamado Schelling Tipping [31], en el se afirma que pequeñas acciones individuales pueden llevar grandes movimientos colectivos. Mejorando este último trabajo, desarrollaron Sugarspace [34], el modelo original es una cuadrícula en donde hay un recurso distribuido (azúcar) y los agentes en cada interacción se mueven a la casilla adyacente con mayor cantidad de ese recurso. Dependiendo de las reglas y parámetros los agentes puede llevar a cabo diferentes acciones.

Con este escenario, de guías y teorías sobre el comportamiento humano desde hace décadas, herramientas tecnológicas y trabajos con simulaciones, surgen las necesidades de mejorar las formas de evacuación en un incendio y la anticipación a ciertas actitudes que se dan en los desastres. Siendo éstas, el motivo principal para realizar este proyecto.

Este trabajo agradece el apoyo y mantenimiento de de la plataforma MASSIS desarrollada dentro del proyecto MOSI-AGIL (S2013/ICE-3019) con la financiación del gobierno de la Comunidad de Madrid y fondos FEDER. El proyecto es liderado por la Universidad Rey Juan Carlos y participan también la Universidad Politécnica y Universidad Complutense. El grupo GRASIA representa a la UCM en este consorcio. El objetivo del proyecto es crear tecnologías que puedan influenciar a los peatones en grandes instalaciones.

1.2. Objetivos

El objetivo principal de este proyecto es la elaboración de un algoritmo o modelo de evacuaciones que siga la teoría de la norma emergente de Turner y Kilian [18]. Para una vez se obtienen los resultados, éstos sean comparados con estudios sociológicos y se determine si son los esperados. El algoritmo debe tener diferentes comportamienos que se dan en situaciones de desastre (pánico, huida, ayuda a evacuación, entre otras) y éstos a su vez tener diferentes tipos de ámbitos (intergrupales, intragrupales, interindividuales e individuales).

Además, para lo consecución de este objetivo:

1. Se ha seleccionado la teoría de la norma emergente, desarrollada por Ralph Turner y Lewis Killian [18], como base para desarrollar el algoritmo.

2. Se ha elegido la aplicación MASSIS para la simulación y creación tanto de agentes con sus comportamientos, como del entorno. MASSIS es una herramienta desarrollada en Java y creada por Rafael Pax.

3. Se ha optado por la representación de un incendio en un centro comercial. Se ha tomado como referencia el Centro Comercial Gran Vía de Hortaleza, situado en Madrid, concretamente la planta baja a excepción del local en el que se encuentra Carrefour junto con otras modificaciones pequeñas.

1.3. Metodología de trabajo.

Para la elaboración de este proyecto ha sido necesaria la creación de tres etapas, motivadas por el estudio de la conducta colectiva y las herramientas para poder simularla, por la planificación del tiempo y trabajo que han sido invertidos en el proyecto y para la correcta presentación de resultados. Estas etapas son de análisis, desarrollo y revisión

crítica.

Análisis.

En esta etapa el tutor facilita gran cantidad de información sobre conducta colectiva y algunos pequeños detalles para enfocar el proyecto. Con esta información como referencia, se investiga sobre autores, teorías, casos, etc. Se estudia la teoría de la norma emergente, se valoran los intereses semejantes que pueden tener los agentes, así como los comportamientos que puedan aparecer, la mayoría de éstos de naturaleza racional [7]. De esta manera, se extraen los elementos más importantes que van a fomar parte de la computerización del modelo. El entorno elegido es un centro comercial en el que se produce un incendio. Surge la necesidad de concretar el algoritmo que se desarrollará, se insiste en enlazar cada comportamiento con la información disponible sobre conducta colectiva.

Desarrollo.

Se empieza a trabajar con la herramienta MASSIS, con ayuda del entorno de desarrollo integrado (IDE) Eclipse. Para facilitar el uso y la compresión de esta herramienta, se opta por la versión de Maven. Entre los requisitos para el correcto funcionamiento de la herramienta, además de un IDE y Maven, también se necesita tener instalada en el pc una versión de Java posterior a la 1.6 y una versión de Ant. Los elementos a parametrizar en MASSIS para este proyecto son tres clases de Java: para los comportamientos de los agentes, para la propiedad del desastre, para diferenciar y contabilizar el número de agentes de cada comportamiento en las simulaciones. También se diseña el tipo de edificio en el se va a desarrollar la simulación mediante el entorno gráfico.

Con este fin, se escoge como períodos dentro de esta etapa:

1. Desarrollar comportamientos individuales con reacción por el desastre con pruebas.

2. Integración de comportamientos anteriores junto a desarrollo de comportamientos grupales e interindividuales con pruebas.

3. Diseño de escenario del desatre en el entorno gráfico de MASSIS con pruebas.

A lo largo de toda esta etapa sigue la comunicación con el tutor por medio de hilos de conversaciones en correo electrónico, en las cuales el tutor resuelve dudas y ofrece sugerencias. En una de estas conversaciones el tutor agrega al autor de MASSIS, Rafael Pax, con la finalidad de solucionar un problema que surge a la hora de que un individuo sea capaz de detectar en un radio definido el incendio. Lo que parecía un error en una de las clases ya implementada en MASSIS, era un error de programación en el proyecto.

Revisión crítica.

En esta fase final del trabajo, se realizan las pruebas con todas las partes elaboradas en la etapa de desarrollo ya integradas, se comparan con estudios de piscología social y sociología. Se redactan conclusiones y se pone fin al trabajo con este documento que intenta recoger la información producida y consultada, de forma detallada, durante todo el proceso.

1.4. Estructura de la memoria.

La memoria de este proyecto está estructurada en secciones, de la siguiente forma:

- Sección 1: Sección actual, en la que se hace una en breve introducción en forma de motivación, se establece el objetivo del trabajo y se detalla la estructura de este documento.

- Sección 2: en esta sección se introducen conceptos que introducen al lector en el campo en el que se va a desarrollar el proyecto. Como son la teoría de la norma emergente y la simulación basada en agentes, entre otros.

- Sección 3: Aquí se introduce el diseño del algoritmo, se razonan los comportamientos utilizados y las interacciones.

- Sección 4: Se explica la implementación del algoritmo, se profundiza en el funcionamiento de los elementos de software del proyecto y se muestra el diseño del escenario para la simulaciones.

- Sección 5: Se presentan los resultados de las simulaciones y se agrupan para compararlos con estudios sociológicos.

- Sección 6: Exposición de conclusiones resultantes de este trabajo y vías para una mejora de éste.

- Sección 7: Exposición de conclusiones resultantes de este trabajo y vías para una mejora de éste en inglés.

- Sección A: Explicaciones del código que no se detalla en las secciones anteriores .

2. Estado del arte

En esta sección se tratan aspectos fundamentales que dan forma al trabajo expuesto en este documento. La base del proyecto es el comportamiento de multitudes, concretamente el comportamiento de multitudes en evacuaciones o en un incendio, debido a esto se detallan estudios y teorías sobre el tema, entre ellas, la teoría de la norma emergente de Turner y Killian [18]. Para conseguir el objetivo final, se ha usado simulación por ordenador. Entre las simulaciones que se pueden llevar a cabo por ordenador, están el vuelo en un avión, las evoluciones de la población de depredadores y la población de presas, el funcionamiento del cerebro humano o el comportamiento de una multitud ante un desastre. Más adelante se detalla que son los sistemas multiagentes, se explica el modelado basado en agentes, que es de donde proviene la aplicación utilizada para simular, MASSIS [42], y se indaga en estudios en los que se ha simulado o se ha dejado unas premisas para la simulación de comportamientos animales o humanos. De este último grupo cabe destacar las dos guías de buenas prácticas desarrolladas por Manuel Fidalgo [43][44] como parte muy importante para la elaboración de este proyecto.

2.1. Comportamiento de multitudes.

Se han realizado algunos estudios de la conducta humana en incendios como el de Cortés, en 1995 [13] afirma que en los edificios públicos, la mayoría de la gente se comporta de forma racional aunque existen conductas inapropiadas debido a factores diferenciales. El autor defiende que uno de estos factores diferenciales es la importancia de los ayudantes no oficiales (líderes espontáneos) en la gestión de la evacuación del incendio antes de que los servicios especializados entren en escena.

Sime en 1994 [14], explica que la ingeniría se basa en el modelo de movimiento

humano, a lo que él llama, “ciencia física”, en esta área se asume que en una situación de peligro y con posibidad de no tener escapatoria, la gente tiene una conducta irracional, de pánico y competición por el acceso a la salida.

Proulx y Sime [15], destacaban, en la evacuación de una estación de metro, la importancia de enviar mensajes directivos a la muchedumbre, estos mensajes tenían que tener contenido sobre el peligro, las conductas a realizar y motivos de dichos comportamientos. Esta información transmitida influyó en que la evacuación tuviera éxito, al originar una evacuación más rapida y con mayor número de comportamientos adecuados.

La psicología multitudinaria, en diversos entornos, también ha sido objeto de estudio, como consecuencia de esto, se han creado algunas teorías que intentan explicar los comportamientos que tienen las multitudes.

Dentro de las teorías clásicas, Sigmund Freud en 1921 [16], defiende que las personas organizadas en multitudes actúan de manera diferente hacia la gente, que las personas no organizadas en multitudes. Cada persona se convierte dentro de la multitud en un ser menos consciente de sus instintos.

La teoría de la convergencia de Milgram y Toch [17], mantiene que el comportamiento de la multitud no es resultado del grupo en sí, es transferido a la multitud por individuos particulares. Por lo que se resume que las personas que se quieren comportar de una manera determinada se unen en grupos.

La teoría de la norma emergente es propuesta por Ralph Turner y Lewis Killian [18] en 1987 con el trabajo de Muzafer Sherif como referencia. En ella, se afirma que el comportamiento social no se puede predecir completamente, pero que las multitudes no son irracionales. El hecho de que los individuos de una multitud compartan intereses, hace que se creen normas de comportamiento de la muchedumbre. Estas pautas pueden ser variables. Por esto, la capacidad de decidir es muy importante. La conducta de la multitud está guiada por las decisiones de los individuos que la forman, pero se rige también por normas que aparecen en el progreso del acontecimiento. Para Turner y Killian los individuos dentro de las multitudes tienen roles diferenciados, algunos actúan como líderes, seguidores, en estado de indiferencia o oponentes. Por ejemplo, un aficionado se levanta, aplaude, canta en un partido de fútbol de su equipo para expresar apoyo y ánimo en un momento en el que considera que es necesario, este comportamiento es seguido por los demás aficionados creando así una norma en el momento. Esta situación expuesta anteriormente como ejemplo, se pudo observar en los partidos de la selección española de

fútbol en el mundial de 2010, como se puede apreciar en la figura 1.

Figura 1. Aficionados de España durante el Mundial de 2010 de fútbol.

2.2. Sistemas Multiagente.

Los sistemas multiagente (SMA) o de inteligencia artificial distribuida (IAD) son una ciencia que abarca a los sistemas que están compuestos por entidades (agentes) a nivel micro [20].

Los agentes tienen una actitud autónoma, operando sin intervención directa de los seres humanos, con algún nivel de control sobre sus acciones y su estado interno. Poseen habilidad social puesto que interactúan con otros agentes, puede que con ser humanos también, por medio de algún lenguaje de comunicación [19]. Los agentes perciben el entorno y responden a los cambios que se producen en él, pero no solo actúan en respuesta al entorno sino que son capaces de mostrar una conducta dirigida a un objetivo tomando la iniciativa, estas dos conductas les hacen poseer tanto actitud reactiva como proactiva [22].

Los SMA tienen gran trascendencia en dominios de investigación y aplicaciones, de tal manera que, pueden ser de gran ayuda para el modelado, diseño, implementación y entendimiento de diversos tipos de sistemas distribuidos. Esto facilita su uso como paradigma de programación para desarrollar sistemas de software distribuidos que se ejecutan en entornos en donde el análisis o control global es muy complicado de alcanzar. También proporciona simplicidad como otra vía para el modelado, respecto al modelado basado en ecuaciones, para la creación y simulación de sistemas que pueden analizarse en elementos menores que se relacionan [21].

- La corriente formal o clásica crea a las entidades con la mayor inteligencia posible, usando definiciones del problema a resolver y encarga el funcionamiento del sistema en ese conocimiento.

- La corriente constructivista crea a las entidades con inteligencia para que con mecanismos de iteracción, el sistema tenga un comportamiento inteligente que puede o no estar diseñado desde el comienzo o diseñado dentro de las entidades. Este comportamiento recibe el nombre de comportamiento emergente.

Hay diferentes metologías para crea SMA. Algunas de las aplicaciones y metologías usadas para los SMA son [23]:

- Vocales, creada por Yves Demazeau y una de las primeras, clasifica a los SMA desde varios puntos de vista, coincidiendo éstos con las vocales: Agente, Entorno, Interacciones y Organización [49].

- MAS-CommonKADS, esta metodología amplia CommonKADS incorporando conceptos de metodologías orientadas a objetos para su aplicación a la producción de SMA [50][51].

- BDI, se inspira en un modelo cognitivo del ser humano. Los agentes utilizan un modelo del mundo, una interpretación de como perciben el entorno [52].

- MaSE, tiene como base el paradigma orientado a objetos y asume que un

agente es únicamente una especialización de un objeto. Los agentes se coordinan con conversaciones y tienen actitud proactiva para alcanzar metas grupales e individuales [53].

- GAIA, su objetivo es obtener un sistema que priorice alguna medida de calidad global. Pretende ayudar al analista para pasar de unos requisitos iniciales a un diseño listo para implementarse [54].

- INGENIAS del grupo GRASIA de la UCM, expande la metología MESSAGE y facilita herramientas para modelar y generar software de sistemas multiagente [55] [56].

El trabajo realizado en este proyecto no se adapta exactamente a una metodología específica, aunque contiene características de varias. Agrupa y ordena conceptos como agente, entorno e interacciones, al igual que con el uso de la metodología de Vocales. De la metodología BDI recoge la idea de que los agentes acumulen información del entorno y a través de esta información modifiquen la visión del mundo que tiene el agente. La forma de plantear el problema desde los requisitos hasta el diseño listo para implementar es parecida a la que se usa en GAIA.

La noción de agente siempre ha sido fundamental en las ciencias sociales. Proveniente de esta ciencia, y como otra vía frente a los modelos tradicionales basados en ecuaciones, ha surgido la microsimulación.

El econometrista estadounidense Guy H. Orcutt [57] expuso que los modelos macroeconómicos de la época no erán válidos, esto se debía a la poca información que aportaban sobre la importancia de las medidas gubernamentales a nivel micro, en las entidades. Orcutt quería mejorar los modelos de simulación añadiendo el nivel micro, para así obtener resultados concisos y útiles para el estudio de los sistemas sociales. Con esta finalidad se añade a las entidades el diseño de comportamientos mediante reglas. Es usada en diversos modelos de ciencias de la salud, tráfico, econometría, sociedades, entre otros. A pesar de sus orígenes, es considerada el origen de los MAB [20][26].

2.2.1. Simulación de comportamientos de animales.

Boid es una herramienta desarrollada por Craig Reynolds en 1986, con el fin de simular el comportamiento de bandadas de aves, aunque también se utiliza con manadas de muchos tipos de animales [27]. El término Boid (Bird-Oid Object) viene de una abreviatura en inglés que significa un objeto que tiene parecido a un pájaro [28]. Esta aplicación trata un comportamiento emergente, su complejidad es creada por la interacción de los agentes individuales, los boids, bajo un conjuntos de normas o reglas simples. En este primera versión de Reynolds las tres reglas aplicadas a los Boids son [29]:

- Separación: Establecer una posición dentro de la bandada en movimiento para evitar el amontonamiento. En la figura 2 se puede apreciar como el boid (representado con un triángulo verde) toma de referencia las distancias con los tres boids que están dentro de su radio (representados por los triángulos azules dentro de círculo gris) para separarse en la dirección que marca la flecha roja.

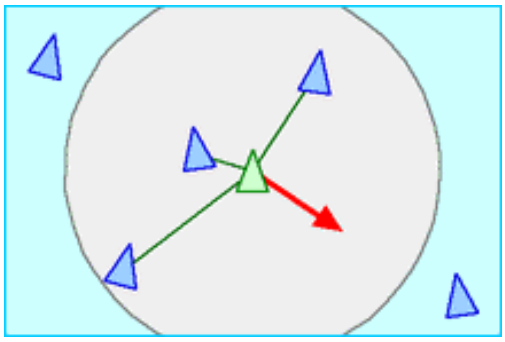


Figura 2. Regla de separación en los Boids.

- Alineación: Poner dirección en común, al mismo punto al que se dirigen los demás boids de la bandada. En la figura 3 se ilustra como el boid (representado con un triángulo verde) toma de referencia las trayectorias de los boids que están dentro de su radio (representados por los triángulos azules dentro de círculo gris) para girarse hacia la izquierda (en la dirección de la flecha roja) pasando de llevar la trayectoria que indica su antena verde a la que indica su antena azul.

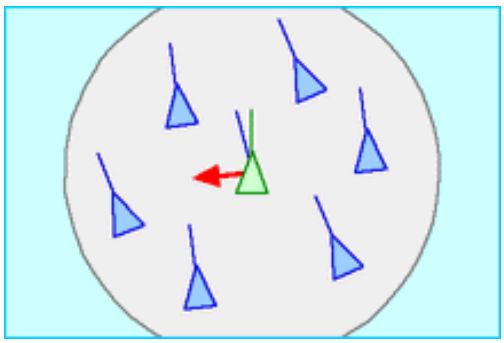


Figura 3. Regla de alineación en los Boids.

- Cohesión: Dirigirse hacia una posición céntrica respecto a los boids más cercanos. En la figura 4 se ilustra como el boid (representado con un triángulo verde) toma de referencia las distancias de los boids que están dentro de su radio (representados por los triángulos azules dentro de círculo gris) para desplazarse hacia la izquierda (en la dirección de la flecha roja) cosiguiendo tener la posición que marca el punto verde, dentro del grupo.

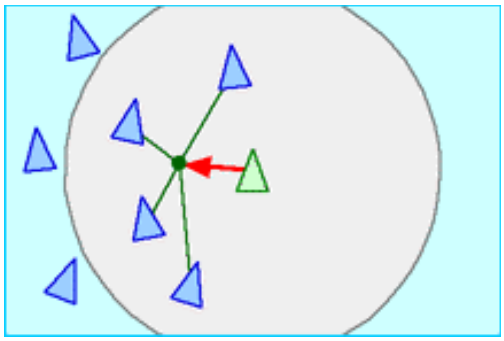


Figura 4. Regla de cohesión en los Boids.

En versiones posteriores se ha ampliado el modelo, añadiendo reglas que lo dotan de mayor complejidad. Estas reglas son la evitación de obstáculos, efectos producidos por el miedo, búsqueda de metas, entre otras. El movimiento de los Boids se puede dividir en caótico por el comportamiento salvaje y la división en grupos, o como ordenado debido a las tres reglas antes mecionadas. La división en bandadas y la reagrupación después de evitar obstáculos se pueden considerar comportamientos emergentes.

El modelo Boids ha servido como inspiración para diversos fines, algunos de éstos son el control directo y estabilización de vehículos, crear animaciones gráficas para películas, programar automáticamente emisoras de radio multicanal de Internet o visualizar información. Ejemplos de estas utilizaciones son los efectos especiales de películas como Batman Returns de 1992 o el Señor de los Anillos, en varias secuelas de la saga [30].

Pese a la gran aceptación de este modelo para movimientos de multitudes, la escasa importancia que se le atribuye al entorno, debido a la no existencia de obstáculos o de fenómenos atmosféricos como el viento, ha hecho que hayan surgido modelos en los que el entorno gane protagonismo.

2.2.2. Simulación de comportamientos humanos.

El trabajo del premio Nobel de Economía de 2005, Thomas Schelling, en el que destacan sus dos libros: *The Strategy of Conflict* y *Micromotives and Macrobehavior* ha tenido gran importancia [31]. Schelling explica un modelo de segregación, que luego adquirió el nombre de *Schelling Tipping Model*. Este modelo defiende que pequeñas tendencias individuales pueden causar grandes tendencias colectivas [32]. Un ejemplo de esto son dos grupos de agentes, los triángulos y los cuadrados, los agentes estarán "descontentos" si menos de un tercio de sus vecinos (otros individuos que están en las ocho posiciones adyacentes) es de la misma clase (triángulos o cuadrados). Estarán "felices" si se da la situación contraria. A partir de una disposición inicial aleatoria, con poblaciones más o menos iguales y con huecos en los que no haya agentes, se mueven a los "descontentos" hasta que la totalidad de los dos grupos sean "felices". Esto da como resultado poblaciones de agentes segregadas en distintas zonas, casi con independencia del nivel en el límite inferior de tolerancia a vecinos "iguales". A pesar de poder encontra una corriente que lleva a la segregación con este modelo, no significa que no influyan más causas u otras corrientes en casos reales [33]. En la figura 5, se puede observar un ejemplo de este modelo de segregación en el que los agentes, tanto cuadrados azules como triángulos amarillos, tienen un límite inferior de tolerancia igual a un tercio, quedando una segragación de cincuenta y nueve porciento en varias poblaciones repartidas por todo el mapa.

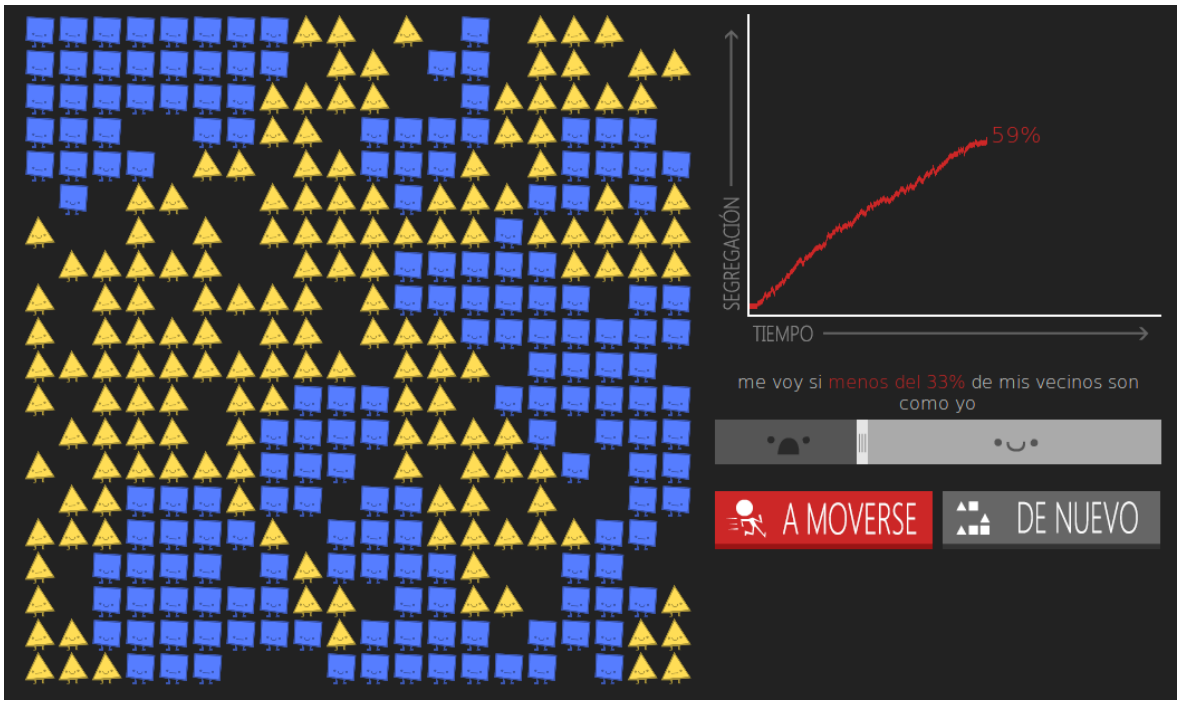


Figura 5. Ejemplo de *Schelling Tipping Model* con límite inferior de tolerancia a vecinos igual a un tercio, segregación final.

A partir del modelo de Schelling y mejorando la simulación *Game of Life* de John Conway, Joshua Epstein y Robert Axtell [34] crearon el modelo de simulación conocido como *Sugarscape*. Los modelos que tienen como base este, cuentan con entidades o agentes, entorno y reglas que dirigen la interacción entre agentes y también con el medio ambiente.

El modelo original consiste en una cuadrícula de 51 celdas de ancho y 51 celdas de ancho, cada celda puede contener diferentes cantidades de azúcar. En cada interacción los agentes se mueven a la celda adyacente con mayor cantidad de azúcar y se la comen. Dependiendo de las reglas definidas y los parámetros introducidos, los agentes pueden realizar acciones como morir, reproducirse, transferir información, entre muchas otras. El azúcar se puede ver como un recurso en un mundo artificial, de esta manera, con los resultados se podrían estudiar fenómenos sociales como la evolución o las herencias de las poblaciones de los agentes. La simulación exacta que aparece en el libro *Glowing Artificial Societies* no siempre es posible recrearla con los mismos resultados.

Con *Sugarscape* como base, se han desarrollado varias aplicaciones de software libre como Ascape, MASON (Multi-Agent Simulator Of Neighborhoods) o Mathematica [35][36][37]. En la figura 6 se muestra un instante en la simulación de la aplicación Sugarscape, en esta ilustración se puede apreciar diferentes tonalidades de amarillo que

representan las cantidades de azúcar, siendo la tonalidad mas oscura donde más hay y la tonalidad más cerca del blanco donde no hay, los agentes son mayoría en las tres tonalidades más oscuras.

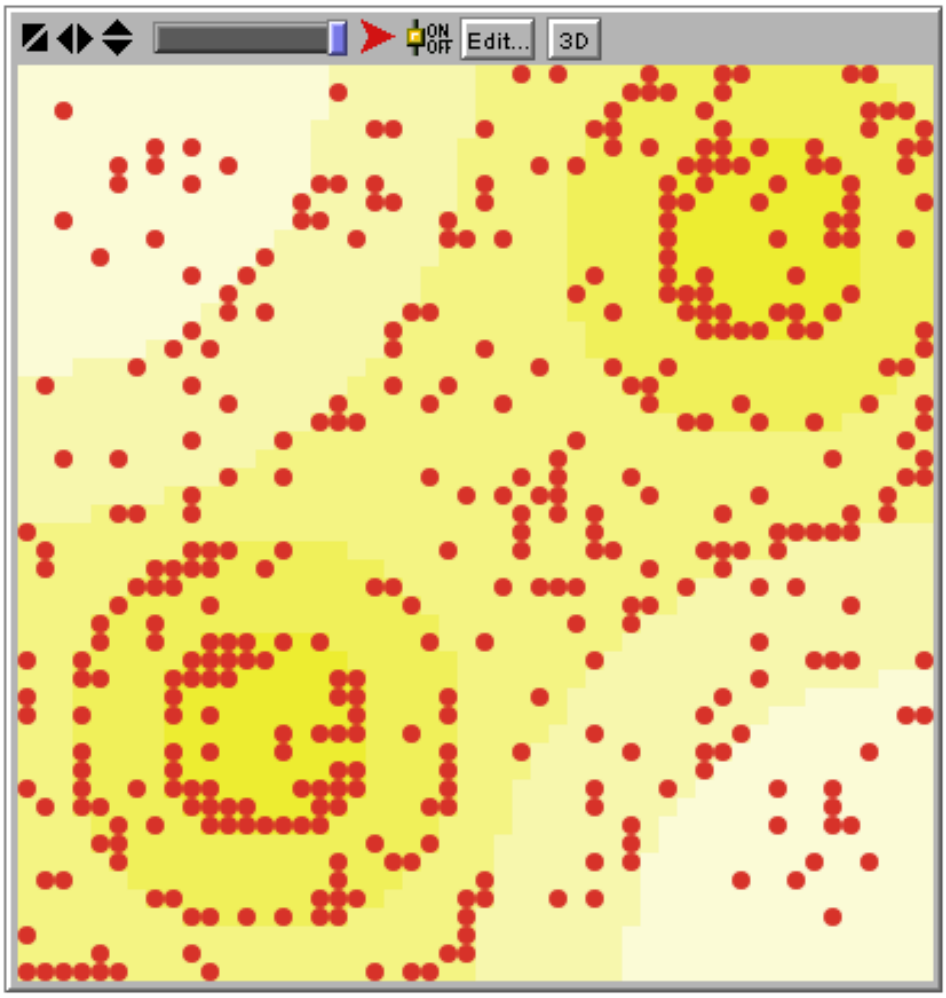


Figura 6. Simulación del modelo *Sugarscape*.

Además de los estudios nombrados anteriormente en este documento sobre evacuaciones en incendios. Canter [38] propuso un modelo general de comportamiento humano en incendios, formado por tres etapas que señalan en que momentos es probable que suceda una alteración en el comportamiento del individuo: la interpretación, la preparación y el acto.

La interpretación se efectua posteriormente a que el individuo perciba las primeras señales de que existe un incendio, puediendo éste malinterpretar o ignorar. Es importante tener en cuenta la actividad previa de la persona como un factor para predecir acciones así como el tipo de señales que captó y la velocidad de reacción ante éstas. La necesidad por

parte del individuo de buscar información mas detallada significa que existe ambigüedad en las primeras señales de incendio.

La preparación tiene lugar cuando el individuo nota humo, tras esto surgen las posibles acciones como dar órdenes a otros, investigar la situación o escapar del lugar. El papel de la persona y el tipo de edificio en el que se encuentra tienen mucha importancia.

El acto es la realización de las tres posibles acciones anteriores que tienen como resultado la evacuación, la lucha contra el incendio, avisar a otros o esperar ayuda. Estas acciones resultantes dependen en gran medida también del papel del individuo, la experiencia en incendios y el tipo de edificio en el que se encuentra.

En este modelo resalta un aumento progresivo en la diversidad de posibles acciones en el transcurso de una etapa a la posterior, debido a esto, las acciones son menos predecibles en las últimas etapas. Al incoporar acciones con comportamientos en incendios, fue propuesto como modelo de sistemas y, al aplicar los resultados recopilados en estudios de incendios reales, contiene cualidades heurísticas [39].

Paulsen [40] trató de resumir el enfoque central sobre comportamiento humano en incendios, afirmaba que independientemente de un ambiente muy estresante, la gente se comporta de forma racional, con tintes de altruismo dentro de los límites que la emergencia ponga a cada individuo respecto a conocimiento, percepciones y acciones. El pánico y las reacciones instintivas no son comunes. Los medios de comunicación y las normas sobre emergencias, a lo largo de la historia, han dado una importancia perjudicial e inconveniente a la reacciones instintivas y de pánico. Existe un estudio empírico creado por Talayero y Aragonés [41] respecto a esta fama en los medios.

Por último, hacer mención a los dos trabajos que sirven como base de este proyecto, estos trabajos son los realizados por Manuel Fidalgo. Se trata de dos guías de buenas prácticas, elaboradas para el Gobierno de España. En ambos estudios analiza la conducta humana ante situaciones de emergencias, pero se diferencian en que en uno de ellos tiene un enfoque individual, en la actitud del individuo [43], mientras que el otro estudio tiene un enfoque global o de multitud, referente a la conducta colectiva [44].

El estudio que está enfocado en la actitud del individuo trata diferentes factores que la influyen, tanto psicológicos como la personalidad, la tolerancia a la frustración, o el liderazgo, como también analiza factores físicos como la edad, sexo o incapacidad. La percepción del desastre por parte del individuo como la forma de actuar una vez lo conoce también son parte de este trabajo, de hecho en la última parte se detallan las variables que influyen en las reacciones y los tipos de reacciones (detallados en la sección 3 de este

trabajo). Se detallan y desgranan medidas preventivas como formación en emergencias o adiestramiento para mejorar las evacuaciones.

Respecto al trabajo de Fidalgo en el que se estudia la conducta colectiva, detalla tipo de multitudes, tipos de espacios en los que se dan las emergencias y conductas que se contangian por la formación de anglomeraciones. Hace un análisis en profundidad de los brotes de pánicos y del fenómeno del contagio mental. También explica las fases que se dan durante el desarrollo de una catástrofe y finaliza de forma parecida al trabajo sobre conducta individual, ofreciendo medidas preventivas para combatir las emergencias.

3. Algoritmo o modelo de evacuaciones.

3.1. Diseño de Algoritmo.

Ante la cantidad de información encontrada en las guías de buenas prácticas de Manuel Fidalgo [43][44], se optó por hacer una criba y utilizar para el diseño del algoritmo la información que se consideró relevante y útil.

De esta manera, tras la selección y fusión de información, las variables finales son:

- Conocimiento de la emergencia: Almacena si el agente se ha percatado del desastre. Hasta la activación de esta variable, o lo que es lo mismo, hasta que el agente tenga conocimiento del desastre, las demás variables no tendrán influencia en el comportamiento.

- Forma de conocer la emergencia: En este parámetro se almacena el método por el cual al agente le llega lo que ha ocurrido. Se diferencian dos formas, en la primera el agente conoce la situación de manera directa y en la segunda conoce la situación de manera indirecta. Un ejempo de conocimiento directo es la visualización del desastre por el individuo. Por otro lado, un ejemplo de conocimiento indirecto es la visualización del comportamiento de otra persona que ya conoce el desastre.

- Conocimiento del lugar: Es el resultado de la unión de las factores: Grado de conocimiento del lugar, tipo o categoría del espacio y existencia de salidas de socorro. Registra si el individuo es consciente de la infraestructura en la que se encuentra.

- Experiencia: Al fusionar los factores, grado de entrenamiento y experiencia anterior en desastres se tiene como resultado este parámetro. Determina la capacidad de liderazgo que tiene el agente, si el agente tiene experiencia será capaz de ejercer de experto y evacuar a otros agentes, si carece de experiencia será seguidor, agente en pánico o agente en huida.

- Por experto: Dentro de las formas de conocimiento de la emergencia indirectas, ya sea por comunicación entre agentes o por visualización de la conducta de un agente por un igual. Se separa al individuo en dos categorías, expertos/líderes y seguidores. Esta variable almacena la categoría del agente que ha facilitado la información sobre el desastre.

En función de los factores considerados, cada agente en la simulación decidirá por una de las siguientes acciones:

- Pánico: El agente conoce el desastre de forma directa y se queda paralizado ante él.
- Huida: El agente huye del desastre hacia una zona segura o la zona más alejada del desastre, con este comportamiento alerta a los demás agentes.
- Ir a ver qué pasa: La entidad se da cuenta de que pasa algo de forma indirecta, se dirige hacia el desastre para analizar lo sucedido.
- Evacuación: Un agente reune y obliga a seguirle a un número determinado de otros agentes y se trasladan a una zona segura o zona más alejada del desastre.
- Seguimiento: Un individuo sigue a otro hasta que este le lleve a un lugar seguro.

Con la finalidad de establecer un vínculo claro e ilustrativo en las relaciones que existen entre las aportaciones de Manuel Fidalgo [43][44] y el algoritmo de evacuaciones se ha creado la tabla 1, en ella se puede observar tanto las relaciones entre los factores estudiados por Fidalgo [43][44] y las variables del algoritmo de evacuaciones, como entre los comportamientos resultantes de ambos trabajos.

Factores en trabajos de Manuel Fidalgo	***Variables en el algoritmo de evacuaciones***
Percepción del riesgo	*Conocimiento de la emergencia (CE)*
Forma en la que se da cuenta el individuo de la emergencia	*Forma de conocer la emergencia (FCE)*
Grado de conocimiento del lugar, tipo o categoría del espacio y existencia de salidas de socorro	*Conocimiento del lugar (CL)*
Grado de entrenamiento y experiencia	*Experiencia (EXP)*
Proceso de evaluación del individuo (evaluación)	*Por experto (PE)*
Comportamientos en trabajos de Manuel Fidalgo	***Comportamientos en el algoritmo de evacuaciones***
Evacuación: salir del lugar y evacuar a otras personas, prevenir a los demás	*Evacuar a grupo de agentes a zona segura o a zona más alejada del desastre*
Alarma, aviso, ir "a ver que pasa"	*Ir a ver qué pasa*
Conmoción, inhibición, estupor	*Pánico*
Agitación	*Huida o fuga hacia zona segura o a zona más alejada del desastre*
Acción de la persona que sigue a otra que evacua o previene a los demás	*Seguimiento*

Tabla 1. Relaciones entre elementos de los trabajos de Manuel Fidalgo y del algoritmo de evacuaciones.

En la búsqueda de relacionar las entradas del sistema (variables) con las salidas del sistema (comportamientos) de una forma clara e ilustrativa y, también con la idea de que la combinación de variables no creara infinidad de casos. Se ha optado por crear la tabla 2 con las variables como datos lógicos o booleanas.

Las variables se representan de la siguiente manera:

- Conocimiento de emergencia (CE) → Si (1). No (0).
- Forma de conocer la emergencia (FCE) → Ajena (1). Propia (0).

- Conocimiento del Lugar (CL) → Si (1). No (0).
- Experiencia (EXP) → Si (1). No (0).
- Por experto (PE) → Si (1). No (0).

Para que la tabla de comportamientos sea efectiva, la variable CE tiene que tener valor 1 siempre. En el caso contrario, el agente seguirá teniendo un comportamiento "normal" y continuará moviéndose por el entorno de forma aleatoria. Una vez se ha mencionado esto, la tabla de comportamientos tiene esta estructura:

CE → 1

FCE	*CL*	*EXP*	*PE*	*Comportamiento*
0	***0***	***0***	***X****	***Pánico***
0	***0***	***1***	***X****	***Evacuar a grupo de agentes a zona segura***
0	***1***	***0***	***X****	***Huida a zona alejada del desastre***
0	***1***	***1***	***X****	***Evacuar a grupo de agentes a zona más alejada del desastre***
1	***0***	***0***	***0***	***Huida a zona segura***
1	***0***	***0***	***1***	***Seguimiento a experto, por el que conoce la situación, a zona segura***
1	***0***	***1***	***0***	***Ir a ver qué pasa y evacuar a grupo de agentes a zona segura***
1	***0***	***1***	***1***	***Evacuar a grupo de agentes a zona segura***
1	***1***	***0***	***0***	***Huida a zona más alejada del desastre***
1	***1***	***0***	***1***	***Seguimiento a experto, por el que conoce la situación, a zona más alejada del desastre***
1	***1***	***1***	***0***	***Ir a ver qué pasa y evacuar a grupo de agentes a zona más alejada del desastre***
1	***1***	***1***	***1***	***Evacuar a grupo de agentes a zona más alejada del desastre***

**** La X significa, en este caso, que cualquier valor que tenga PE no tiene importancia debido a que la forma de conocimiento es directa, FCE = 0.***

Tabla 2. Relaciones entre variables y comportamientos.

A modo de resumen de toda esta información de relaciones entre variables y comportamientos en el algoritmo, se ha creado la figura 7. En esta figura se puede observar a un agente representado por un triángulo blanco, a la izquierda está la lista de variables del algoritmo (unidas al agente por flechas rojas) que influye en el comportamiento del

agente y a la derecha la lista de comportamientos (unidos al agente con flechas verdes) que tienen los agentes como consecuencia de la combinación de los valores introducidos en cada variable.

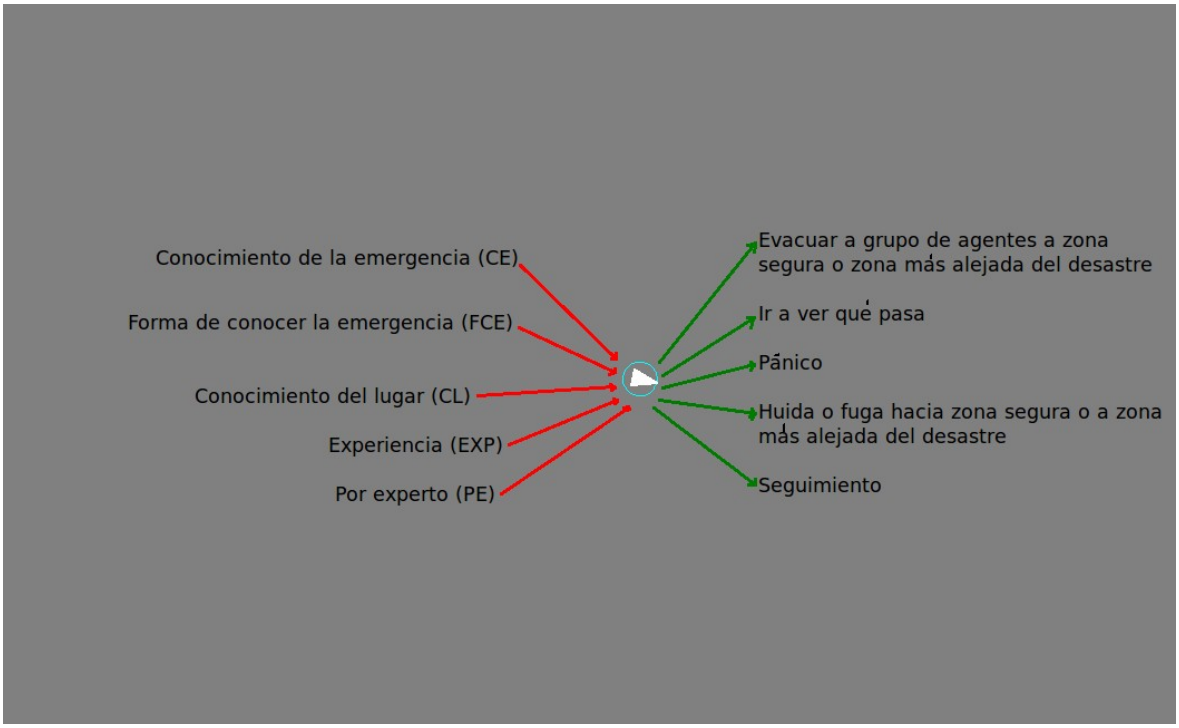


Figura 7. Relación entre variables (flechas rojas entrantes al agente) que afectan al comportamiento y comportamientos (flechas verdes salientes al agente) que puede tomar el agente.

Las variables y los comportamientos han sido conseguidos en su totalidad, como ya se ha mencionado con anterioridad en este documento, de los estudios de Manuel Fidalgo [43][44], pero la forma de relacionar los estados de las variables con el comportamiento resultante ha sido una decisión subjetiva del autor de este trabajo, basándose en el conocimieno adquirido con el tiempo dedicado al estudio de los comportamientos humanos en situaciones de emergencia. En los siguientes apartados se detalla cada comportamiento de la tabla.

En el momento de valorar la reacción de pánico para el algoritmo, se tuvo en cuenta que las personas que no han tenido experiencia de hallarse ante una situación de peligro pueden experimentar desde una actitud de calma hasta verdadero pánico, EXP = 0. Evitar señales exageradas de peligro, que pueden ser germen de pánico para el individuo. La señal

de peligro mas exagerada que pueda encontrase el individuo es conocer el desastre de forma directa, o lo que es lo mismo, FCE = 0. Para que se desencadene este tipo de conducta, el acento no hay que situarlo en el número de personas, sino en la escasez de salidas o vías de evacuación. Se puede concluir que es posible que ante la escasez de conocimiento sobre el lugar en el que se encuentra el individuo, este puede creer que hay escasez de salidas o vías de evacuación, CL = 0. El pánico provoca la paralización por completo del agente, pudiendo observarse en la simulación que se queda en la posición en la que percibió el incendio. La situación de pánico dura hasta que un experto obliga a seguirle al individuo.

La huida se da en dos tipos. En el primer tipo, cuando el individuo conoce de forma directa la emergencia (FCE = 0), tendrá un perfil parecido al individuo que entra en pánico (inexperto, EXP = 0) pero al conocer el recinto (conoce el lugar, CL = 1) y por lo tanto las vías de escape, se podrá ver durante la simulación al agente desplazándose hacia la zona más alejada del desastre.

En el segundo tipo, el agente también inexperto (EXP=0), conoce el desastre mediante una persona que huye (forma indirecta de conocer el desastre, FCE=1 y no es por experto, PE=0), de esta forma entra en escena la teoría de la norma emergente [18], con la situación similar de los dos agentes, con interés prioritario por salvarse ambos. El agente que conoce el desastre por el agente que huye, al no conocer la infraestructura en la que se encuentra (conoce el lugar, CL = 0), huirá hacia una zona segura que no será siempre la más alejada del desastre. El agente se moverá en dirección a una zona alejada y segura, que en algunos casos coincidirá que es la más alejada pero en la mayoría de los casos no se dará esa opción.

Los dos comportamientos están respaldados por la percepción y evaluación sobre el contexto y los recursos adaptativos propios para superar cualquier daño o pérdida, buscando el bienestar que se detallan en el trabajo de Fidalgo [43][44].

Ir a ver que pasa es la acción en el que un experto (EXP=1) conoce el desastre por un agente inexperto (forma de conocer la emergencia, FCE=1 y por un agente que no es experto, PE=0) que se encuentra huyendo. Con esto, el agente se dirige hasta el lugar donde se encuentra el incendio. Según las guías, para el entrenamiento ante situaciones de emergencia [18] se tiene en cuenta el liderazgo, por lo tanto una persona con entrenamiento (en este algoritmo llamado experiencia, EXP) es un líder. Indican las guías también, que un

líder deberá atender a la intensidad del riesgo. Se puede deducir que tendrá que analizar el desastre de forma directa. Este comportamiento es complementario con el de evacuación.

En la evacuación, el agente experto (EXP=1) conoce el desastre de primera mano o por otro experto, o bien, tuvo conocimiento por otro inexperto y despues de ir a analizar (ir a ver que pasa) ha decidido empezar a evacuar (da igual de la formas que se de cuenta el agente experto del desastre y si es por otro experto o no, FCE=X y PE=X, siempre tendrá este comportamiento). Para esto el agente intenta contactar con tres inexpertos que no tengan conocimiento del incendio o que estén en estado de pánico. Para ello se mueve aleatoriamente por todo el entorno. Una vez consiga a tres agentes inexpertos que le sigan procede a evacuar de dos formas.

La primera forma la realiza si conoce el lugar (CL=1), en ese caso evacuará a sus tres seguidores hacia la zona más alejada del desastre y permanecerán allí.

De la segunda forma, sin conocimiento del lugar (CL=0), el agente experto evacuará a los tres seguidores hacia una zona segura que en muchos casos no será la zona más alejada del desastre.

En la información consultada en el trabajo de Fidalgo [43][44], el líder dirige y sirve de modelo de comportamiento.

El seguimiento se ha vinculado a los agentes que han pasado por la conducta de pánico (forma de conocer la emergencia es directa, FCE = 0, sin conocimiento del lugar, CL = 0, y sin experiencia, EXP = 0), o, a aquellos que no son expertos y se enteran del desastre por un agente experto o líder (la forma de conocer la emergencia es indirecta, FCE = 1, no importa si conocen el lugar, CL = X, sin experiencia, EXP = 0, y conocen el desastre por un agente experto, PE = 1).

En ambos casos, ya sea el del agente inexperto que sale del pánico por el agente experto o el agente inexperto que conoce el desastre por el agente experto, el agente inexperto sigue al agente experto (que le has sacado del pánico o les ha comunicado el desastre) hasta que éste les lleva a una zona segura o la zona más alejada del desastre.

Para los agentes que han experimentando pánico y según se recoge en una de las guías, los líderes pueden impedir, cortar o desacelar una situación de pánico, para luego, dirigirlos y canalizar hacia vías de evacuación. Estas dos últimas acciones de liderazgo, extraídas del trabajo de Fidalgo [43][44], y llevadas a cabo por el experto, sirven para respaldar la decisión de adjudicar este comportamiento a todos los agentes inexpertos que

conocen la emergencia por expertos.

3.2. Multitud e interacciones.

El entorno en el que se realiza la simulación es un centro comercial, por lo tanto, se trata de un espacio cerrado de grandes dimensiones. En este tipo de lugares, se dan concentraciones de personas en un estado de aparentemente calma y poco sentido de unidad, este tipo de concentraciones se clasifican en multitudes casuales. Debido a la aplicación de la teoría de la norma emergente [18], en esta simulación aparecen intereses comunes que acercan a los individuos de la multitud, por ejemplo, escapar y ponerse a salvo o evacuar a otras personas. También como afirma esta teoría, coincidiendo con los comportamientos diseñados en este trabajo, los agentes tienen papeles distintintos. Existen líderes, también llamados expertos, e inexpertos, éstos se clasifican en seguidores, fugados o en huida y en estado de pánico.

Con la investigación sobre reacciones de multitudes en incendios, tanto en espacios cerrados como en espacios abiertos, se han obervado casos de gente que se queda petrificada hasta otros en los que la gente que, gracias a su rápida respuesta, consigue salvarse y poner a salvo a varias personas, pasando por comportamientos extremos como lanzarse desde una ventana de un edificio en llamas.

Para representar el algoritmo diseñado y resumir la información sobre variables y comportamientos se han creado dos diagramas de flujo. En ellos se puede apreciar la importancia de los valores de la variables PE, FCE y CL en los comportamientos adoptados por los agentes. El inicio de los dos diagramas (el oval de conocimiento de emergencia) es este debido a que antes de conocer la emergencia (o con CE=0), el comportamiento de todos los agentes es similar, tienen movimiento aleatorio por todo el entorno, por lo tanto se ha empezado desde el supuesto que los agentes en los diagramas ya conocen el desastre (CE=1). El hecho de que sean dos diagramas, uno para los comportamientos de un experto y otro para los del inexperto, se ha hecho para poder facilitar la compresión y que no quedara un diagrama enorme con exceso de información que no se pueda enteder, esta división supone la asignación de experiencia (EXP=1) en el

diagrama del agente experto y de la falta de experiencia (EXP=0) en el diagrama del agente inexperto. Respecto a los comportamientos están directamente relacionados con las acciones finales de ambos diagramas. En cierta manera, el modelo que defiende este algoritmo de evacuaciones, se puede representar con varios agentes pero de solo dos tipos, estos dos tipos son agentes expertos y agentes inexpertos, que dependiendo de las variables que tienen que ver con la forma de conocer el incendio (Conocimiento de la emergencia CE, Forma de conocer la emergencia FCE y Por experto PE) y de las variables que no cambian durante la simulación (Experiencia EXP y Conocimiento del lugar CL), los agentes actuarán de una forma u otra (dentro de los comportamientos del diagrama de cada tipo). El siguiente diagrama, indica los comportamientos que tiene, en el algoritmo propuesto, un agente experto o líder, tomando como punto de partida el conocimiento del desastre.

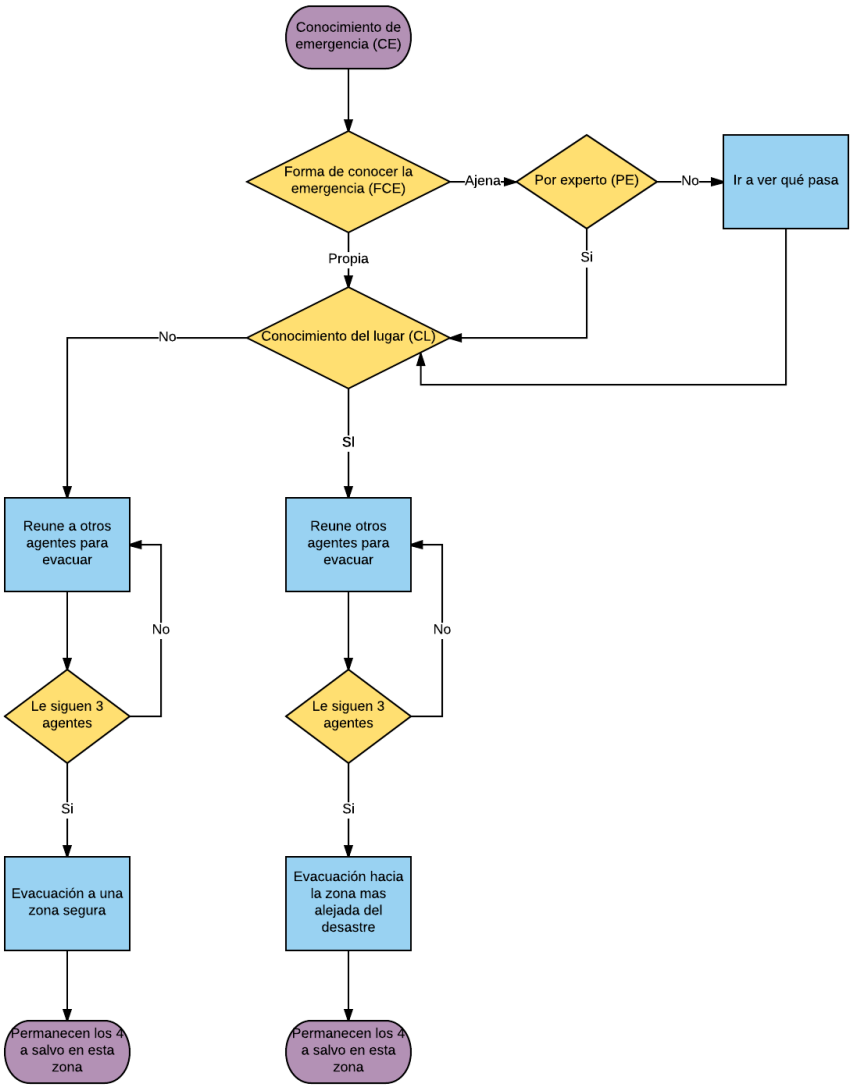


Figura 8. Diagrama de flujo para un agente experto en el algoritmo propuesto.

Este segundo diagrama, indica los comportamientos que tiene, en el algoritmo propuesto, un agente inexperto, con conocimiento de emergencia como acción de comienzo.

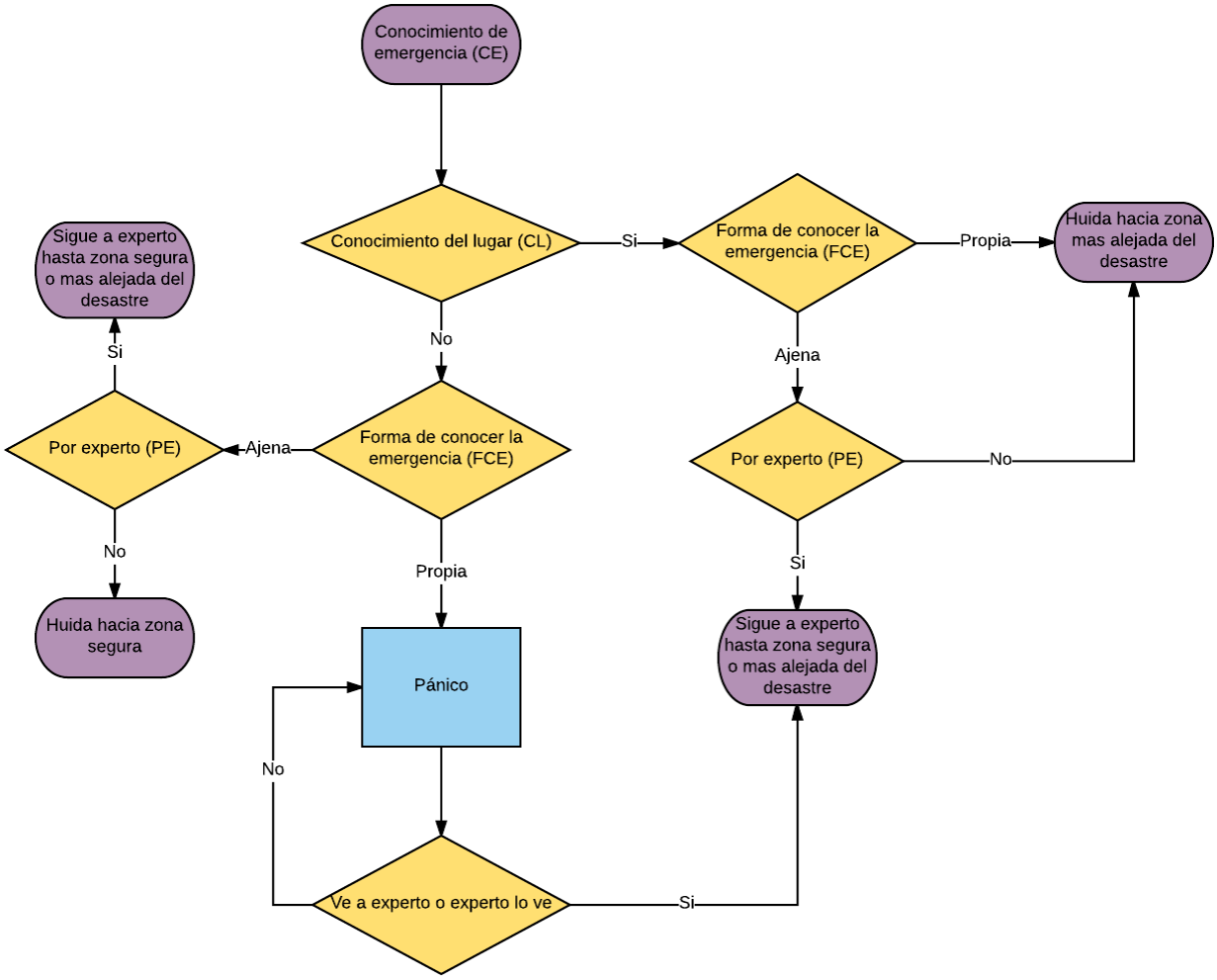


Figura 9. Diagrama de flujo para un inexperto en el algoritmo propuesto.

4. Implementación de algoritmo y pruebas en las simulaciones.

Para conocer y prácticar con la herramienta, MASSIS dispone de tutoriales que aparecen en la página web de la aplicación. Estos tutoriales comienzan por crear un proyecto de MASSIS usando Maven (herramienta de software para la gestión y construcción de proyectos Java [45]) y finalizan con la simulación de un rescate entre robots en una estación espacial, pasando por la definición de un comportamiento simple o de un entorno más complejo y varios agentes.

La forma general de proceder para la implementación del algoritmo de evacuaciones ha sido:

1. Crear un escenario con el editor de entorno de MASSIS y dotar a los elementos de éste, concretamente a agentes y desastre, de las clases que definan sus comportamientos (AgentHighLevelController y ObjectHighLevelController respectivamente) así como de las variables expuestas en la sección 3.1.

2. Se opta por seguir el mismo modelo que se indica en los tutoriales para crear el algoritmo. Para ello, se procede a crear cinco clases:

- AgentHighLevelController: Contiene los diferentes comportamientos de los agentes.

- SimulationWithUILauncher: Esta clase se encarga de ejecutar el simulador.

- ObjectHighLevelController: Se utiliza para dar a un elemento la característica de desastre.

- EnvironmentEditor: Es responsable de iniciar el editor del entorno.

- DisasterSimulationLayer: Contiene el código para diferenciar y contabilizar el número agentes de cada comportamiento en las simulaciones.

3. Se elige un número de población de agentes y un desastre determinado, en este caso un incendio, para luego instanciar las clases asociadas a cada elemento, con vistas ya al escenario final de la simulación.

4. Se lanzan varias simulaciones, en las que se anotan y comparan sus resultados.

4.1. Inserción de valor para las variables.

Cada elemento del editor Sweet Home 3D con complementos MASSIS tiene algunos metadatos asociados. Para editar y agregar los metadatos de un elemento, primero se selecciona el elemento y luego se escoge la opción del menú: Tools → Add Metadata. Al realizar esta acción, aparece un panel que, dependiendo del elemento que se haya seleccionado, puede contener varios de los siguientes metadatos:

- ID: Identificador único, se recomienda no modificarlo, todos los elementos del entorno poseen uno. Tiene un valor númerico entero y positivo. Parte de los procesos internos de MASSIS.

- IS_OBSTACLE: Indica si el elemento es un obstáculo. Todos los elementos en el entorno, incluidos los agentes, son obstáculos. Esta variable, activada, es responsable, por ejemplo, de que los agentes no atraviesen paredes cuando calculan la ruta mínima de un punto a otro . Puede tomar los valores verdadero (true) o falso (false). Parte de los procesos internos de MASSIS.

- IS_DYNAMIC: Señala si el elemento tiene capacidad de movimiento durante la simulación. En este trabajo solo tendran capacidad de movimiento los agentes. Toma los valores verdadero (true) o falso (false). Parte de los procesos internos de MASSIS.

- CLASSNAME: Es la referencia a la clase que implementa el comportamiento del agente o a la clase que indica que es un desastre. Se asigna el valor tutorialfollower.myfirstmassisproject.AgentHighLevelController en caso de ser un agente o tutorialfollower.myfirstmassisproject.ObjectHighLevelController en caso de ser un desastre. Parte de los procesos internos de MASSIS pero con valores de este proyecto.

- DISASTER: Indica si el elemento es un desastre. Se asigna a los objetos llamas y humo. Puede tomar los valores verdadero (true) o falso (false). Añadida para la elaboración del algoritmo de evacuaciones.

- KNOWDISASTER: Se asigna a todos los agentes y es el equivalente en la implementación a la variable Conocimiento de emergencia (CE), nombrada en el diseño del algoritmo. Se ajusta a los valores verdadero (true) o falso (false). Añadida para la elaboración del algoritmo de evacuaciones.

- KNOWPLACE: Se añade a todos los agentes y es semejante a la variable Conocimiento del lugar (CL) en el diseño del algoritmo. Toma los valores verdadero (true) o falso (false). Añadida para la elaboración del algoritmo de evacuaciones.

- METHODKNOWLEDGED: Se asigna a todos los agentes y corresponde al parámetro Forma de conocer la emergencia (FCE) en el diseño del algoritmo. Adquiere los valores verdadero (true) o falso (false). Añadida para la elaboración del algoritmo de evacuaciones.

- EXPERIENCE: Se añade a todos los agentes y es semejante a la variable Experiencia (EXP) en el diseño del algoritmo. Puede tomar los valores verdadero

(true) o falso (false). Añadida para la elaboración del algoritmo de evacuaciones.

- STATE: Se asigna a todos los agentes. Señala si el agente se encuentra siguiendo a alguien, en el caso de los agentes inexpertos, o si se entera del desastre de forma directa o indirecta, en el caso de los agentes expertos. Se utiliza para la obtención de números totales de cada comportamiento durante la simulación. Se ajusta los valores verdadero (true) o falso (false).

Subrayar que byExpert, pese a existir, es una variable interna en cada agente, que comienza a 0 pero que tomará un valor diferente dependiendo de si el agente conoce el desastre por un experto, en el apéndice de final de documento se explica en detalle esta variable. En la figura 10 se muestra el proceso para rellenar los metadatos en el entorno.

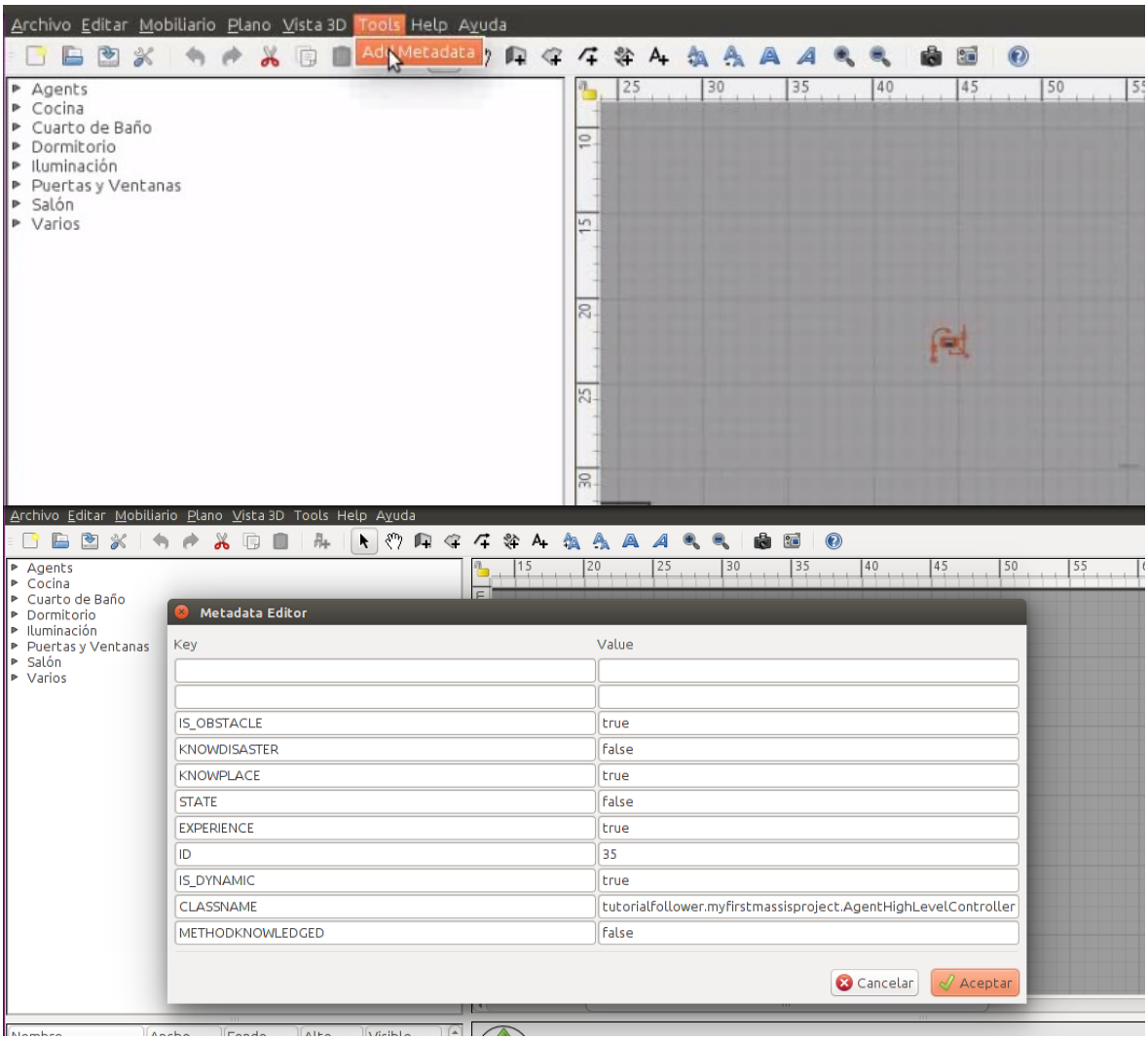


Figura 10. (**Arriba**) Selección en el menú de la opción Tools →Add Metadata, dentro del editor Sweet Home 3D con complementos MASSIS. (**Abajo**) Panel de metadatos y valores al concluir la inserción.

4.2. Implementación de Algoritmo.

Las clases que se encargan de definir los comportamientos son AgentHighLevelController y ObjectHighLevelController, que heredan de la clase HighLevelController. Aunque la clase ObjectHighLevelController es bastante sencilla, es de vital importancia para el método del agente que reconoce un desastre en un rango. Este proceso obtiene, como se explica en el apéndice detalladamente (situado en el final de este documento), los agentes que hay dentro de un rango. Gracias a ello, el desastre es tratado como un agente, con la posibilidad de recoger sus metadatos y encontrar el valor de DISASTER.

AgentHighLevelController (también detallada en el apéndice) tiene como objetivo agrupar todos los comportamientos de los agentes en una misma clase y asignar estos comportamientos, según la combinación de valores de los atributos de cada agente. Estos valores son modificados y accedidos por medio de métodos internos.

Para facilitar la compresión de como se han trasladado las ideas de los diagramas (figuras 8 y 9 de la sección 3.2) a código. A continuación se procede a explicar qué métodos de la clase AgentHighLevelController hacen posibles las acciones de estos diagramas en MASSIS:

- Conocimiento de emergencia: El agente se encuentra en un movimiento aleatorio desde el comienzo de la simulación, provocado por moveRandomly(). A cada paso se verifica dentro de este método si, el desastre se encuentra dentro del rango de visión del agente mediante DisasterRange(range) o, si hay otro agente en su rango de visión que sepa de la existencia del desastre por medio de ByExpertRange(range). En caso afirmativo de alguna de las dos condiciones anteriores, el agente pasaría a conocer el desastre y a almacenar la forma de conocimiento (propia o ajena) con el uso de los métodos setKnowDisaster(boolean) y setMethodKnowledged(boolean). En caso de ser un agente inexperto que conoce el desastre de forma propia o conoce el desastre por un agente inexperto, también almacenará la localización del incendio accediendo a los métodos getLocation() del incedio o getDisasterLocation() del agente inexperto. A partir de este paso entrarían en escena los métodos InexpertBehaviour() o ExpertBehaviour().

- Conocimiento del lugar: Dentro de los métodos InexpertBehaviour() o ExpertBehaviour(), que son los que se encargan de asignar los comportamientos para los agentes inexpertos y expertos respectivamente, se controla si el agente conoce el lugar con el uso del método isKnowPlace(), esta verificación es clave cuando el agente ya conoce la existencia del incendio y tiene que tomar un comportamiento.

- Forma de conocimiento: De la misma manera que Conocimiento del lugar, es ejecutada dentro de InexpertBehaviour() o ExpertBehaviour() y verifica si el agente ha conocido el incendio de forma propia o ajena por medio de isMethodKnowledged(), esta condición es importante cuando el agente conoce el desastre y debe responder con otra conducta.

- Por experto: Ejecutada por los mismos métodos (InexpertBehaviour() o ExpertBehaviour()) y con la misma importancia que los dos anteriores apartados para reaccionar a la existencia del incendio, isByExpert() comprueba si el conocimiento sobre el incedio fue adquirido a través de un agente experto o de un agente inexperto. Esta comprobación se hace cuando la Forma de conocimiento tiene valor ajena ya verificado.

- Ir a ver qué pasa: El agente se desplaza hacia al incendio hasta que éste entra en su rango. Esta acción es ejecutada por el método whatIsUp(Location) en los agentes expertos que previamente han conocido el punto en el que se encuentra el incendio mediante un agente inexperto. La localización objetivo es usada en whatIsUp(Location) por medio de getDisasterLocation().

- Reune otros agentes para evacuar: Se lleva a cabo por un agente experto que recorre aleatoriamente el entorno mediante el método followMe(), examinando en cada “paso” si dentro del rango del agente se encuentra un agente inexperto que no conoce el incendio o en estado de pánico, en caso afirmativo pasaría a ser seguidor del agente experto mediante el método followMeInexpert(range), esto último se consigue con el paso del objeto del agente experto con el método setFollowTarget(agent) del agente inexperto.

- Le siguen 3 agentes: La primera condición que se comprueba en cada “paso” del agente experto en la acción de reunir a otros agentes para evacuar es que tenga menos de 4 seguidores. Esta comprobación se lleva a cabo con el valor de la variable numberOfFollowers, que se incrementa dentro del método followMeInexpert(range), dentro del agente experto, cada vez que un agente inexperto le sigue. También es posible que un agente inexperto (en pánico o que no conoce el incendio) “vea” en su rango a un agente experto que conoce el incendio, antes de que éste último se percate. En este caso, el futuro seguidor comprueba que el experto tiene un número de seguidores menor de 4 e incrementa el número de seguidores del experto dentro de followExpert(range).

- Se dirigen hacia una zona segura o huida hacia zona segura: El agente se dirige hacia una posición alejada con el uso del método escapeToSafeZone(). Este método elige, al azar, una posición en una habitación (todos los espacios son tomados como habitaciones en MASSIS, incluidas las calles de alrededor del centro comercial) de la mitad del vector (ordenado de habitación mas cercana al punto 0,0 del entorno, hasta la mas lejanada) en la que no se encuentra la habitación en la que está el incendio, una vez escogido este punto, el agente se desplaza hasta llegar a él.

- Se dirigen hacia la zona más alejada del desastre o huida hacia zona más alejada del desastre: El agente se dirige hacia una posición en la habitación más alejada del desastre mediante el método escapeToFarthestZone(). Este método selecciona un punto dentro de la primera habitación del vector, en caso de que la habitación se encuentra en la seguda mitad del vector. En caso de que la habitación con el incendio se encuentra en la primera mitad del vector, el método escoge un punto en la última habitación. Con el objetivo seleccionado, el agente se mueve hasta llegar a él.

- Pánico, Ve a experto o experto lo ve: Es la acción en la que el agente inexperto permanece inmóvil al estar en el rango del incendio. El agente está alerta de si algún experto que sabe del incendio y tiene menos de 4 seguidores, entra en su rango para seguirlo, con el uso de followExpert(range). También se encarga de revisar si algún agente experto le ha “dicho” que le siga mediante el método

iAmFollowing(agent). Este método verifica que agent es la referencia de objeto del experto a seguir.

- Sigue a experto hasta zona segura o mas alejada del desastre: El agente inexperto tiene la referencia de un agente experto, ya sea porque la ha cogido o porque se la ha dicho el agente experto. El agente pasa a seguir al experto, hasta que éste llega al punto en la zona segura o mas alejada del desastre. Esto es posible con el método iAmFollowing(agent), que provoca que el agente se mueva a la posición del agente experto (que también está en movimiento) con agent.getLocation().

Se ha intentado resumir y usar poco más que los nombres de los métodos de AgentHighLevelController y ObjectHighLevelController que hacen posible las acciones de los diagramas de la sección 3.2. Entre los métodos nombrados se encuentran los que son responsables de la comunicación entre agentes en diferentes casos: DisasterRange(range), ByExpertRange(range), followMeInexpert(range), followExpert(range). Estos se encargan de encontrar en un rango o distancia a otros agentes, y así se obtiene las referencias a los objetos agentes o desastre que rodean al agente, consiguiendo estos métodos con las referencias poder seguir a ese objeto o modificar valores o su movimiento. De todas formas, como ya se ha dicho a lo largo de esta sección. **Los métodos se explican detalladamente en el apéndice final de este documento.**

4.3. Movimientos en la simulación.

En total son cinco movimientos los que se pueden diferenciar en las simulaciones, estos movimientos son: movimiento aleatorio, reunir otros agentes para evacuar, ir a ver qué pasa, dirigirse o huir hacia zona segura, dirigirse o huir hacia zona más alejada. Para no condensar esta sección demasiado, **las explicaciones detalladas en los métodos y las figuras que faltan, se encuentran en el apéndice final de este documento.**

Movimiento aleatorio: Los agentes escogen, al azar, un punto y una vez llegan a esa localización, vuelven a repetir la acción hasta que se de una condición de cambio de conducta. Todos los agentes ejecutan este movimiento hasta que conocen el desastre. En la figura 11 se puede apreciar este movimiento en dos agentes representados con triángulos blancos y sus trayectorias representadas con líneas rojas. En la figura 12 se ilustra el código del método **moveRandomly()**.

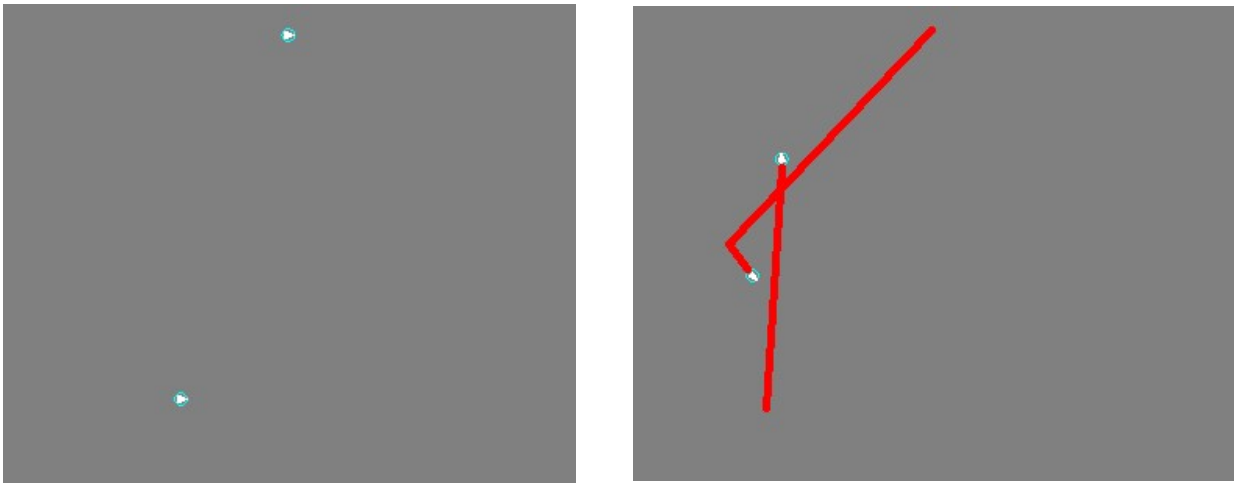


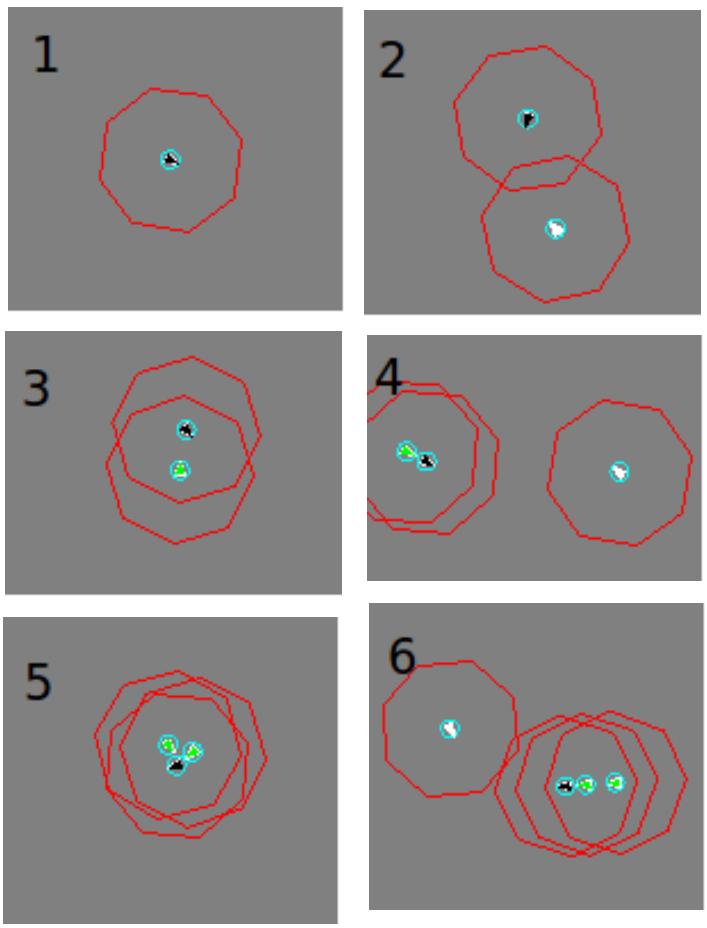
Figura 11. (**Izquierda**) Posición inicial de agentes que se desplazan aleatoriamente. (**Derecha**) Mismos agentes después de varios pasos, en rojo la trayectoria de cada uno. Los cambios de trayectoria son cambios en el objetivo.

```
void moveRandomly(){
    if (this.currentTarget == null) {
            Random rnd = ThreadLocalRandom.current();
            Location agentLocation = agent.getLocation();
            Floor agentFloor = agentLocation.getFloor();
            List<SimRoom> roomsInFloor = agentFloor.getRooms();
            int numberOfRooms = roomsInFloor.size();
            int rndRoomIndex = rnd.nextInt(numberOfRooms);
            final SimRoom rndRoom = roomsInFloor.get(rndRoomIndex);

            Location randomLocation = rndRoom.getRandomLoc();
            this.currentTarget = randomLocation;
    }

    ApproachCallback callback = new ApproachCallback() {
        @Override
        public void onTargetReached(LowLevelAgent agent) {
            currentTarget = null;
        }

        @Override
        public void onSucess(LowLevelAgent agent) {
            if(DisasterRange(search_range)){
                setKnowDisaster(true);
                setMethodKnowledged(false);
            }

            else if(ByExpertRange(search_range) > 0){
                setKnowDisaster(true);
                setMethodKnowledged(true);
            }
        }

        @Override
        public void onPathFinderError(PathFinderErrorReason reason) {
            Logger.getLogger(AgentHighLevelController.class.getName())
                .log(Level.SEVERE,
                        "Error when approaching to {0} Reason: {1}",
                        new Object[] { currentTarget, reason });
        }
    };

    this.agent.approachTo(this.currentTarget, callback);
}
```

Figura 12. Método moveRandomly de la clase AgentHighLevelController.

Reunir otros agentes para evacuar: Este movimiento empieza con un deplazamiento aleatorio por parte de un experto que sabe que hay un incendio. Este experto contacta o es contactado por agentes inexpertos (depende de que agente se de cuenta primero de que el otro está en su rango) que no saben que existe el desastre o que están en pánico. Los métodos que hacen posible este movimiento son **FollowMe()** en el caso del experto (es el método por el cual el experto se mueve aleatoriamente, dentro de éste, busca a los agentes dentro de su rango, para cuando encuentra un inexperto en pánico o que no conoce el desastre, le indica su posición a través de una referencia a su objeto y el inexperto le siga), e **iAmFollowing(agent)** en el caso de los seguidores (método por el cual, el inexperto o futuro seguidor toma la referencia al objeto del líder que es agent, y le sigue).

En la figura 13 se representa un ejemplo de esta acción, el agente experto representando en negro, se encuentra agentes que no saben del desastre representados en blanco, una vez se enteran de éste (pasan a ser representados de color verde) por medio del agente le siguen. Cuando el agente experto tiene tres seguidores, inicia la evacuación hacia una zona segura o hacia la zona más alejada del desastre. El octógono rojo que rodea a cada agente representa el radio de visión de cada uno de ellos. El código de iAmFollowing(agent) se puede observar en la figura 14.



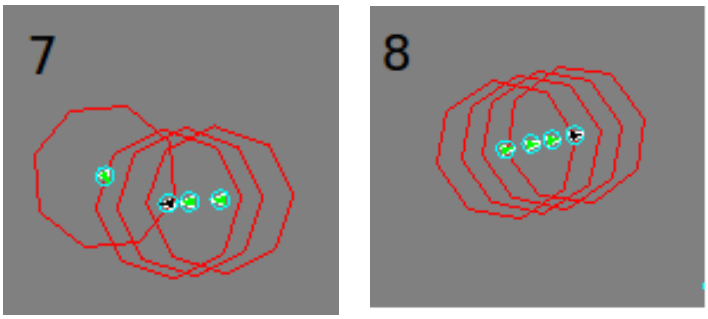
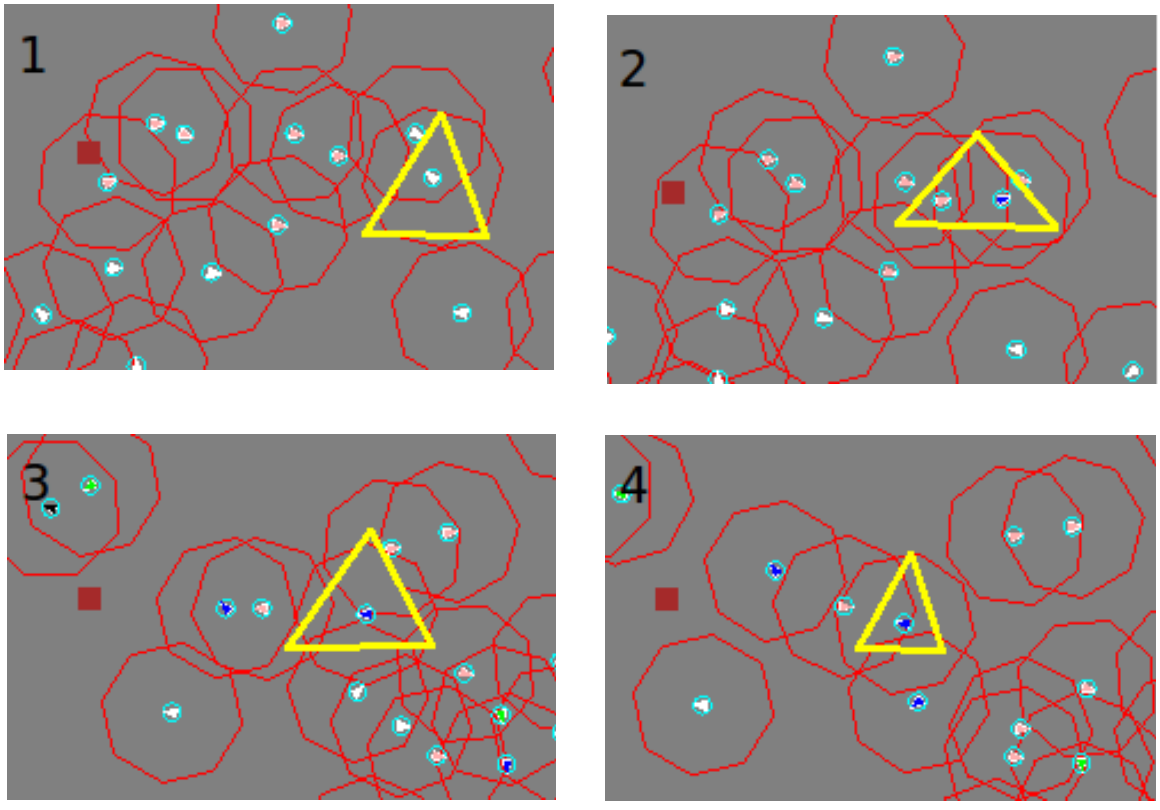


Figura 13. Desplazamiento de un agente experto representando en negro, que encuentra agentes que no saben del desastre representados en blanco, una vez se enteran de éste por medio del agente (pasan a color verde) lo siguen, cuando llegan a tres seguidores, el grupo inicia la evacuación.

```
void iAmFollowing(LowLevelAgent agent){
    if (agent != null) {
        setState(true);

        this.agent.approachTo(agent.getLocation(),
          new ApproachCallback() {

            @Override
            public void onTargetReached(LowLevelAgent agent) {

            }

            @Override
            public void onSucess(LowLevelAgent agent) {

            }

            @Override
            public void onPathFinderError(PathFinderErrorReason reason) {
                Logger.getLogger(AgentHighLevelController.class.getName()).log(
                  Level.SEVERE,
                  "An error occurred when approaching to {0}. Reason: {1}",
                  new Object[] { followTarget.getLocation(), reason });

            }
          });
    }
}
```

Figura 14. Método iAmFollowing de la clase AgentHighLevelController.

Ir a ver qué pasa: Un agente experto conoce el punto exacto donde existe un incendio por medio de un agente inexperto en huida. El experto para asegurarse que la información transmitida por el inexperto es correcta y “analizar” el incendio, se deplaza hasta éste hasta que entra en su rango de visión. En la figura 15 se representa un ejemplo de esta acción, en ella un agente experto (representado por un triángulo blanco dentro de un triángulo mayor amarillo) conoce el desastre por un agente inexperto en huida entre otros inexpertos (representados por los triángulos rosas), de esta forma y con la información aportada por los inexpertos en huida (El agente experto accede al DisasterLocation que tienen almacenado el agente inexperto por el que conoce el desastre), se dirige hacia la localización exacta del desastre (representado con un cuadrado marrón, . El método que ejecuta este movimiento es **whatisUp(Location)**, su código se puede observar en la figura 16.



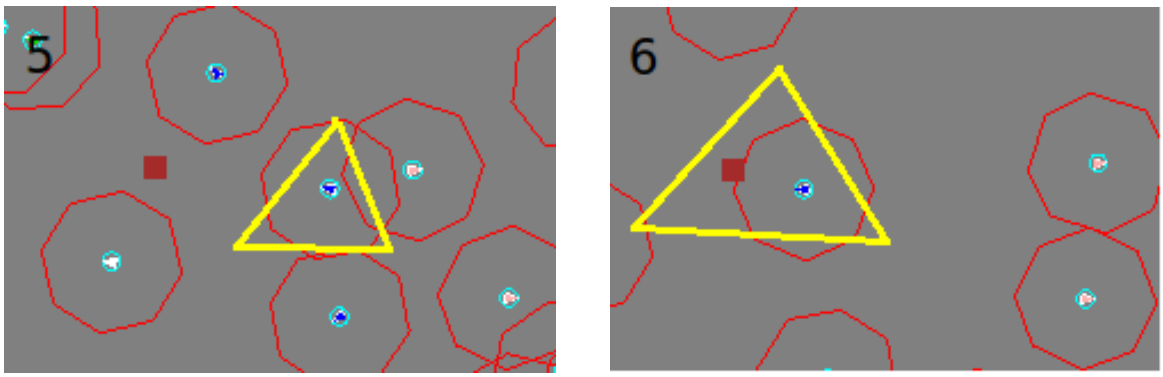


Figura 15. Desplazamiento de un agente experto (dentro de un triángulo amarillo) hasta la posición del desastre (representado por un cuadrado marron). Primero, el agente se mueve aleatoriamente, se encuentra con un grupo de agentes inexpertos huyendo (representados por triángulos rosas) hacia una zona segura, en el que uno le transfiere la posición exacta del desastre, el experto se dirige hacia el desastre (representado por un cuadrado marrón) hasta que entra en su rango de visión.

```
private void whatisUp (Location target){

    ApproachCallback callback = new ApproachCallback() {
        @Override
        public void onTargetReached(LowLevelAgent agent) {

        }

        @Override
        public void onSucess(LowLevelAgent agent) {
            if(DisasterRange(search_range)){
                setDisasterLocation(null);
            }
        }
```

Figura 16. Parte del método whatisUp de la clase AgentHighLevelController.

Evacuar o huir hacia zona segura: El agente, ya sea un experto seguido por tres inexpertos o un inexperto, se dirige a un punto en una habitación, al azar, que se encuentra en la mitad del vector (ordenado de habitación mas cercana al punto 0,0 del entorno, hasta la mas lejanada) en la que no está la habitación que contiene el incendio. En la figura 17 se aprecia como un agente inexperto, después de conocer el desastre y encontrándose en la en la habitación 0, se desplaza primero hasta la habitación 1 para llegar por último a la

habitación 3 que es una zona segura pero no la más alejada del desastre. El método que ejecuta este movimiento es **escapeToSafeZone()**, su código se puede ver en la figura 18.

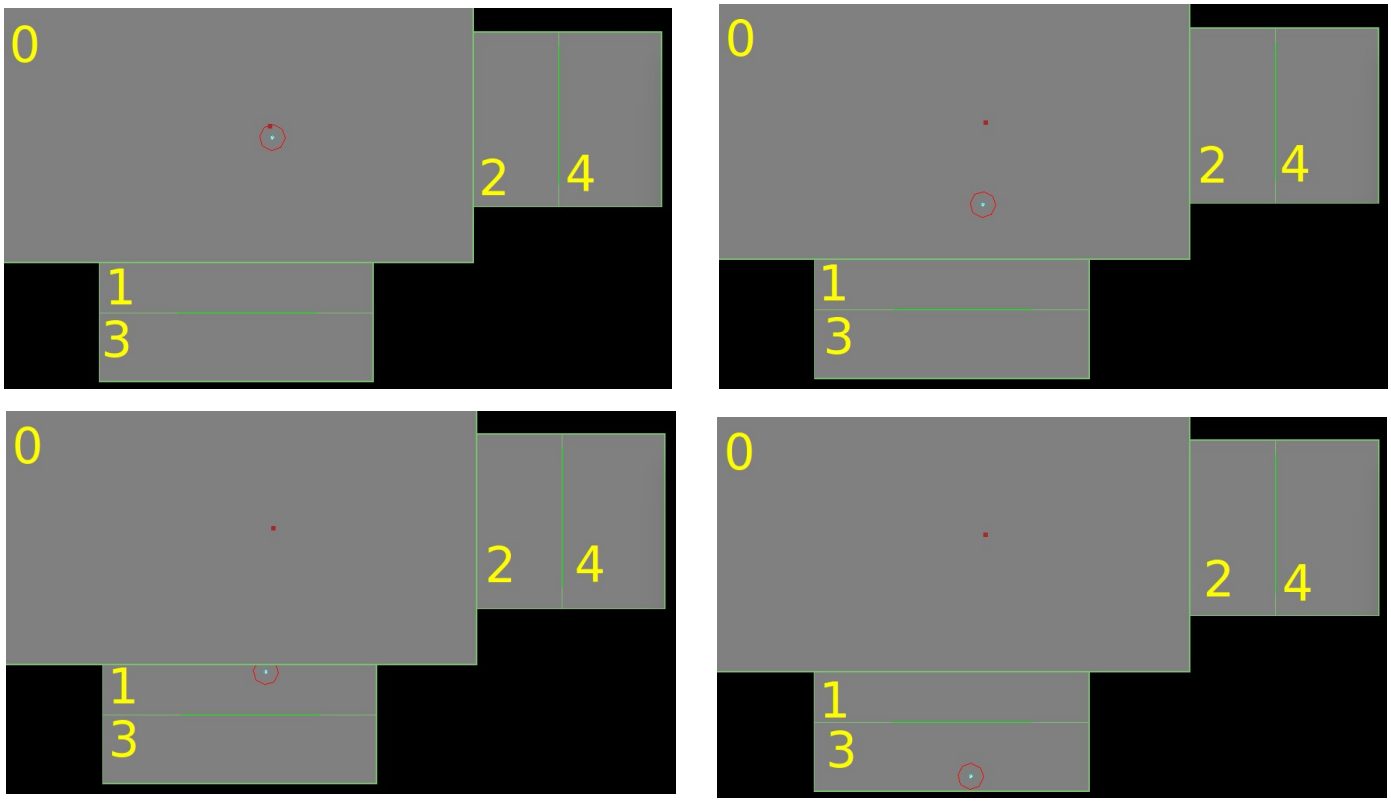


Figura 17. Desplazamiento de un agente inexperto hasta una zona segura, después de haber conocido el desastre (representado por un cuadrado marron). En amarillo los índices que ocupa cada habitación cuando se ordenan por distancia.

```
void escapeToSafeZone(){
    int aux = this.agent.getRoom().getRoomsOrderedByDistance().size()/2;
    int last = this.agent.getRoom().getRoomsOrderedByDistance().size()-1;
    Random rnd = new Random();
    int dummy = 0;

    if (destination == null){
        for(int i = 0; i < last+1 ; i++){
            if(this.agent.getRoom() == this.agent.getRoom().getRoomsOrderedByDistance().get(i)){
                if(i<=aux){
                    dummy = rnd.nextInt((last)-(aux+1))+(aux+1);
                }
                else{
                    dummy = rnd.nextInt((aux-1));
                }
            }
        }
        destination = this.agent.getRoom().getRoomsOrderedByDistance().get(dummy).getRandomLoc();
    }

    ApproachCallback callback = new ApproachCallback() {
        @Override
        public void onTargetReached(LowLevelAgent agent) {
            stop = true;
            setDisasterLocation(null);
        }
```

Figura 18. Parte del método escapeToSafeZone de AgentHighLevelController.

Evacuar o huir hacia zona más alejada del desastre: El agente efectúa un movimiento parecido al explicado en el apartado anterior pero su objetivo es un punto en la habitación más alejada de la habitación que contiene el incendio. En la figura 19 se representa el movimiento de un agente inexperto que conoce el desastre (representado por un cuadrado marrón) e inicia el movimiento hasta la habitación 4, pasando previamente por la número 2, la habitación 4 es la mas alejada respecto a la posición 0,0 en el entorno. El método que ejecuta este movimiento es **escapeToFarthestZone()**, su código se puede ver en la figura 20.

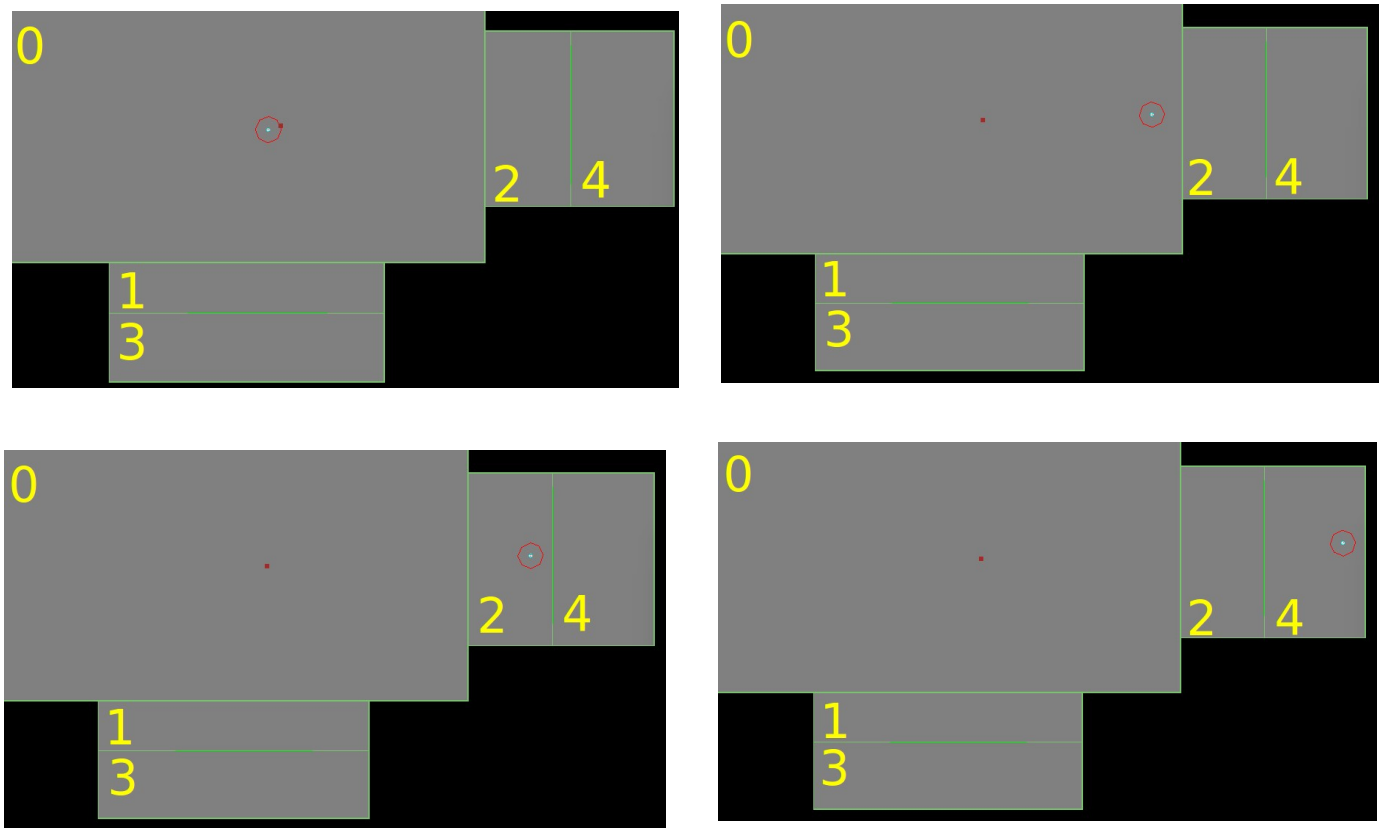


Figura 19. Desplazamiento de un agente inexperto hasta la zona mas alejada, después de haber conocido el desastre (representado por un cuadrado marron). En amarillo los índices que ocupa cada habitación cuando se ordenan por distancia.

```
void escapeToFarthestZone(){
    int aux = this.agent.getRoom().getRoomsOrderedByDistance().size()/2;
    int last = this.agent.getRoom().getRoomsOrderedByDistance().size()-1;

    if(destination == null){
        for(int i = 0; i < last+1 ; i++){
            if(this.agent.getRoom() == this.agent.getRoom().getRoomsOrderedByDistance().get(i)){
                if(i<=aux){
                    destination = this.agent.getRoom().getRoomsOrderedByDistance().get(last).getRandomLoc();
                }
                else{
                    destination = this.agent.getRoom().getRoomsOrderedByDistance().get(0).getRandomLoc();
                }
            }
        }
    }

    ApproachCallback callback = new ApproachCallback() {
        @Override
        public void onTargetReached(LowLevelAgent agent) {
            stop = true;
            setDisasterLocation(null);
        }
```

Figura 20. Parte del método escapeToFarthestZone de AgentHighLevelController.

4.4. Visualización de las simulaciones.

Aunque la opción 3D para las simulaciones puede ser mas realista, la vista en 2D ofrece un mejor enfoque en el análisis y la depuración. Además, la visiualización en 2D permite la creación de capas definidas por el usuario. Se pueden añadir varias capas a una misma visualización. Las capas pueden ser activadas o desactivadas en cualquier punto de la ejecución. La vista 2D en MASSIS viene con varias capas predeterminadas, útiles en muchas ocasiones y de fácil uso. En la figura 21 se observa el panel Layer dentro de la ventana para la representación de la simulación, dentro de este panel se encuentran las capas y la posiblidad de habilitarlas o deshabilitarlas.

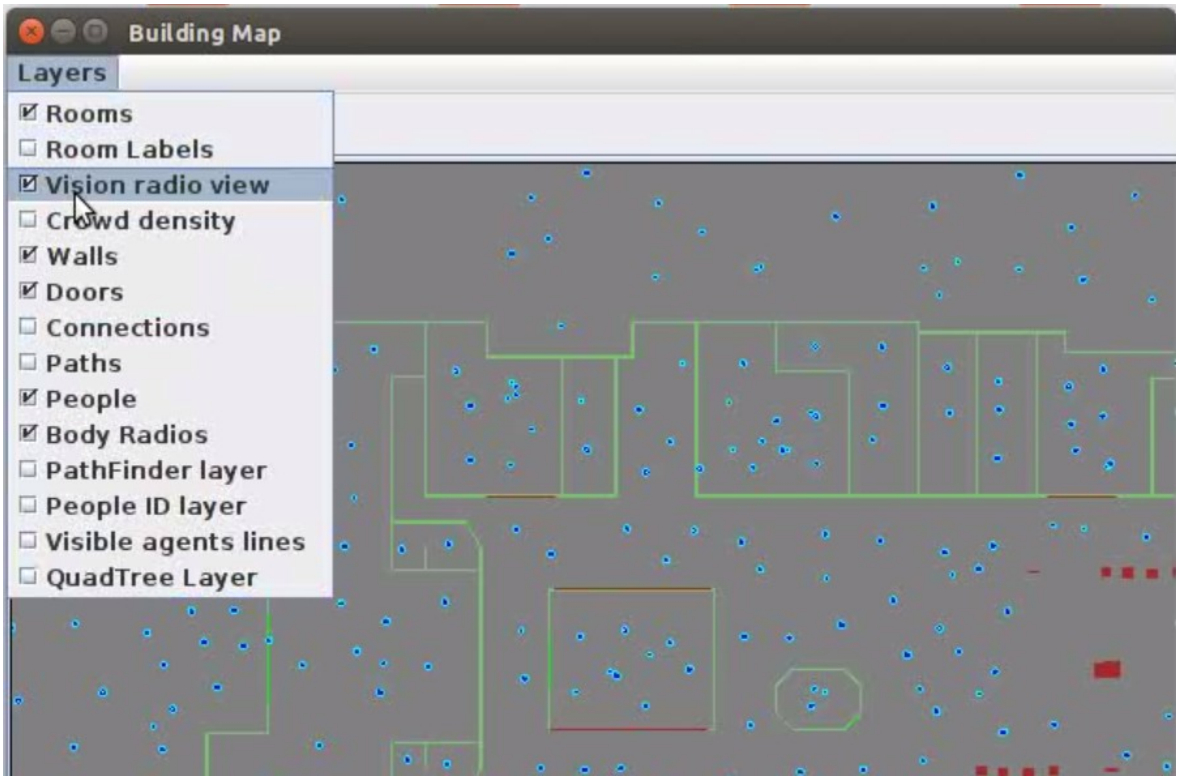


Figura 21. Ventana de simulación 2D de MASSIS con menú de capas predeterminadas desplegado.

Sin embargo, estas capas predeterminadas, no solucionan dos problemas que se presentan a la hora de llevar acabo las simulaciones del algoritmo. Uno de estos problemas es diferenciar los comportamientos desde el primer momento y comprobar las variaciones durante la simulación. Y el otro problema se trata de contabilizar los totales de cada tipo de comportamiento para poder hacer análisis y sacar estadísticas.

Para resolver estos dos problemas, se ha optado por implementar en el archivo DisasterSimulationLayer y añadir a la simulación una capa personalizada, esta capa se llama DISASTER SIMULATION AGENTS. Con esto, se soluciona el primer problema, pero el segundo todavía estaba por resolverse. Para evitar dar con la solución definitiva, se añadieron unas variables que actúan de contadores cada vez que esta capa pinta a los agentes según los comportamientos que tengan. Cada comportamiento tiene un color y están asignados como se puede apreciar en la Tabla 3.

Comportamiento	***Color***
Experto conoce el desastre de forma directa	***Negro***

Experto conoce el desastre de forma indirecta	***Azul***
Inexperto que huye o está paralizado	***Rojo***
Seguidor	***Verde***
Agente que desconoce el desastre (Movimiento aleatorio)	***Blanco***

Tabla 3. Relaciones entre comportamientos y colores en la simulación 2D.

De esta misma manera se creó también un marcador que imprimiera por consola cada dos pasos el número total de agentes que había de cada comportamiento. El formato de este marcador es el que se puede ver en la figura 24.

Estas dos soluciones fueron fundamentales para validar las simulaciones, debido a que la visiualización pasó a ser mas completa. De esta forma, no es necesario emplear otros métodos más tediosos, como mostrar los IDs de cada agente y su posición en el entorno, para saber cuantos agentes expertos conocen de primera mano el desastre. Esta capa, como muestra la figura 22, consigue la diferenciación de comportamientos y agentes durante la simulación de una manera visual y simple, gracias a los colores en los agentes que se puede ver en la ventana del entorno y al marcador de totales que se puede ver abajo.

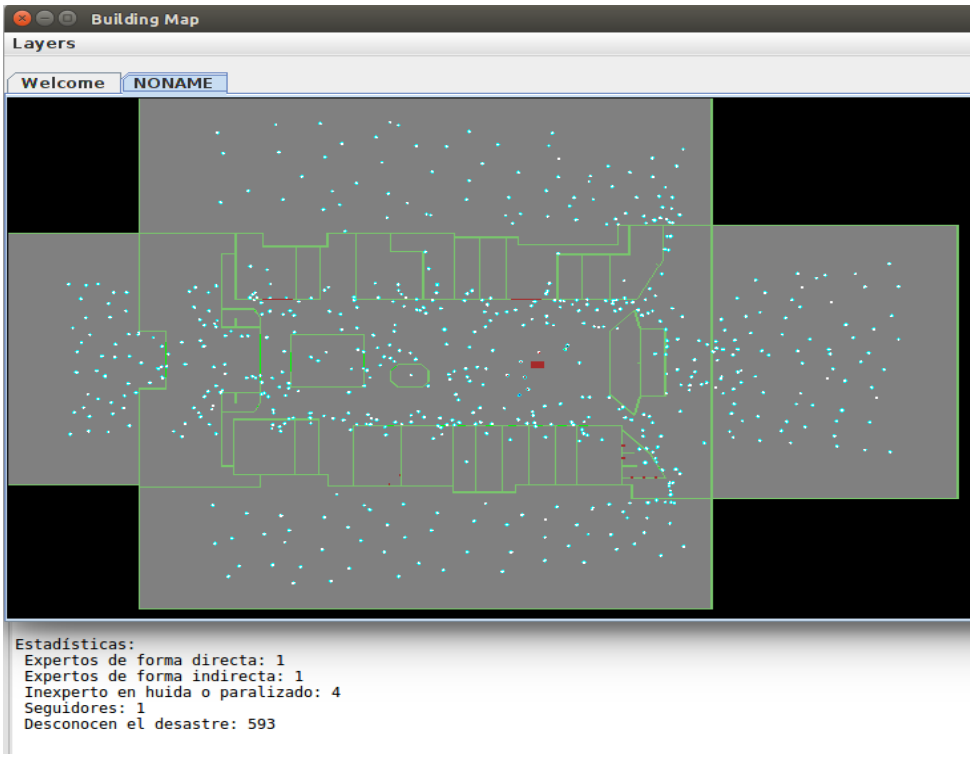


Figura 22. Simulación del algoritmo en 2D y marcador de totales en la parte inferior izquierda.

5. Caso de estudio.

El edificio tomado como referencia para simular el algoritmo ha sido el Centro Comercial Gran Vía de Hortaleza. Es un centro comercial situado en Madrid, concretamente en el distrito de Hortaleza, entre las calles de Arequipa y Ayacucho, con entrada principal en la calle Gran Vía. Cuenta con más de 40 tiendas, repartidas en dos plantas y dedicadas a sectores como alimentación, deportes y ocio, hogar, salud y belleza, entre otros [46]. Se ha elegido la primera planta y tras no encontrar planos con información exacta, se ha optado por utilizar la herramienta de medición de Google Maps. Como se puede ver en la figura 25, gracias a esta herramienta se ha podido saber que la superficie de la primera planta, excluyendo un hipermercado, es de 8500m² aproximadamente. En la figura 23 se puede obsevar la medición del perímetro de la planta baja del Centro Comercial Gran Vía de Hortaleza (excluyendo uno de sus locales) con la heramienta Google Maps y el total en metros de la superficie medida, en una ventana centrada en la imagen, abajo.

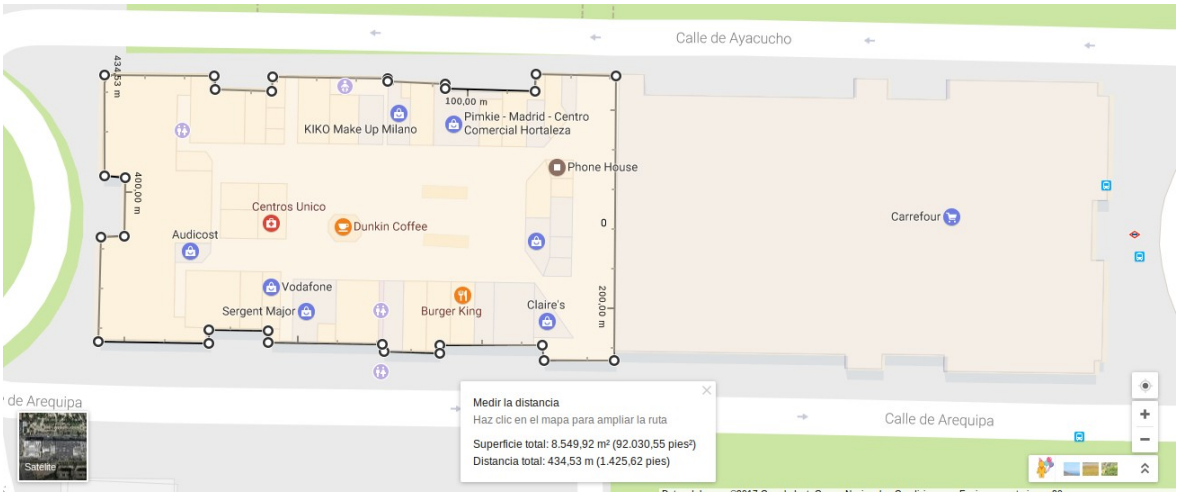


Figura 23. Medición de Centro Comercial Gran Vía de Hortaleza con Google Maps.

Con estos datos, y después de medir todos los elementos situados en el interior de esta superficie. Se ha diseñado con el entorno gráfico de MASSIS el edificio. Para ello se

han importado nuevos modelos de agentes, fuego, humo, plantas, coches, entre otros. También ha habido modificaciones en la estructura del diseño obtenido de la medición. Se han eliminado las escaleras mecánicas, la entrada del hipermercado (que no se ha diseñado) ha pasado a ser una entrada al centro comercial y el número de locales o cuartos se ha reducido, dejando un total de 29 espacios en todo el diseño, 25 dentro del centro comercial y 4 que simulan las calles a las que da el centro comercial. En la figura 24 se pueden apreciar las vistas en 2D y 3D del escenario final con todos los elementos.

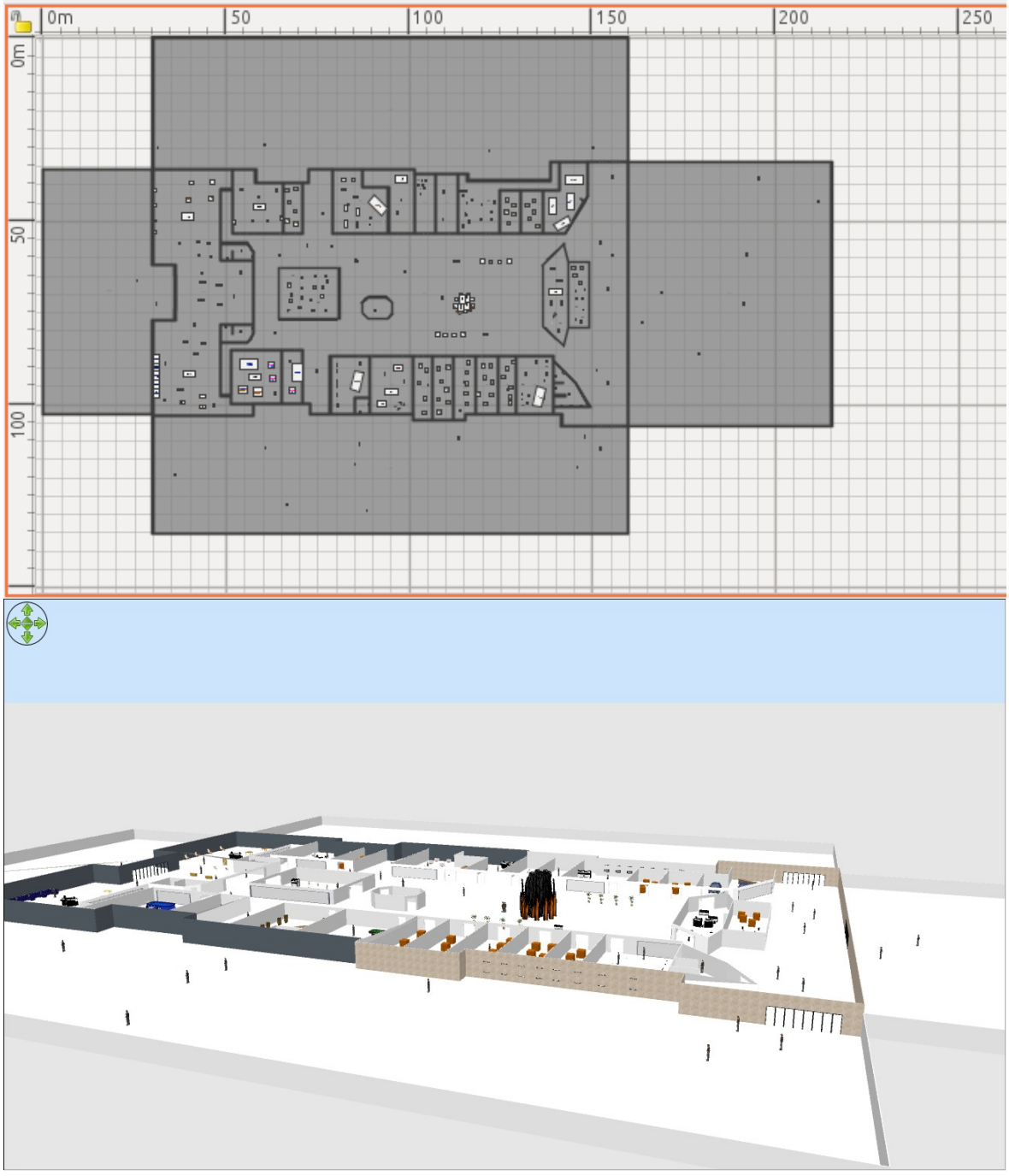


Figura 24. (**Arriba**) Diseño de edificio final en el entorno gráfico de MASSIS en 2D. (**Abajo**) Diseño de edificio final en el entorno gráfico de MASSIS en 3D. En el centro de

la imagen se puede apreciar el fuego y humo.

Para maximizar el rendimiento de la simulación se han suprimido la mayoría de elementos pertenecientes al mobiliario. Han permanecido en el escenario los siguientes elementos: agentes, fuego, humo y estructura del edificio. En la figura 25 se puede apreciar en 2D, el estado final del escenario una vez suprimida la mayoría de los elementos del mobiliario.

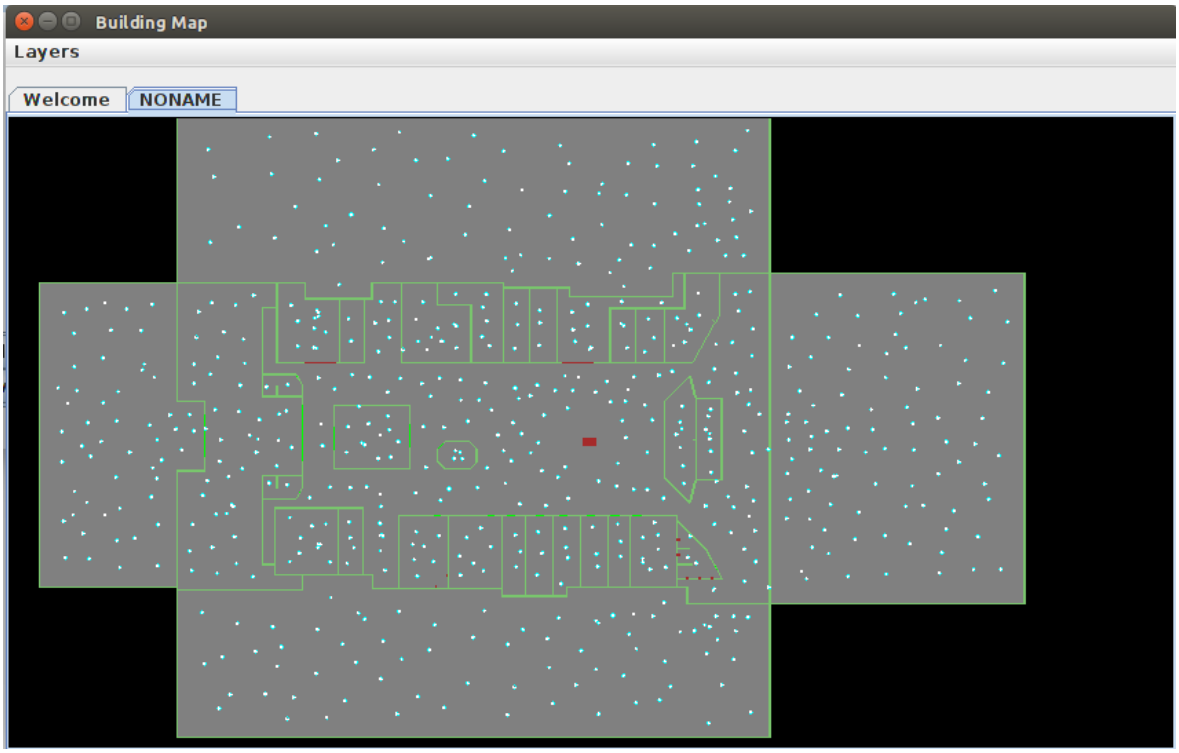


Figura 25. Imagen inicial de simulación después de eliminar mobiliario.

Un total de 600 agentes, divididos en grupos, como se puede apreciar en la tabla 4, según las propiedades que permanecen invariables (KNOWPLACE y EXPERIENCE) durante toda la ejecución de la simulación. Los metadatos usados y a los que se les ha asignado valor son EXPERIENCE, KNOWPLACE, KNOWDISASTER, METHODKNOWLEDGED, STATE, DISASTER y byExpert, además, han sido utilizados, los metadatos que vienen en la herramienta por defecto y son esenciales para las funciones de la herramienta (ID, IS_DYNAMIC, IS_OBSTACLE y CLASSNAME) . Los agentes con EXPERIENCE con valor 1 siempre serán expertos cuando conozcan el desastre,

mientras que los que tenga valor a 0 serán seguidores (en función de si conocen el desastre por un experto, por lo que METHODKNOWLEDGED valdrá 1 y byExpert 2), agentes en pánico (en función de si conocen el desastre de primera mano, por lo que METHODKNOWLEDGED valdrá 0, byExpert 0 y KNOWPLACE 0) o agentes en huida/fugados (en función de si conocen el desastre por un inexperto, por lo que METHODKNOWLEDGED valdrá 1 y byExpert 1, o de primera mano pero conocen el lugar, por lo que METHODKNOWLEDGED valdrá 0, byExpert 0 y KNOWPLACE 1). En el caso de KNOWPLACE con valor 1, es igual a que los agentes conozcan el lugar, tanto los agentes expertos como inexpertos, evacuarán o huirán respectivamente a la zona más alejada del desastre. En caso de que está variable tenga valor 0, los agentes expertos evacuarán y los inexpertos huirán (cuando se de el comportamiento de huida) a zonas seguras pero muchas veces no serán la más alejadas del desastre. Los agentes están distribuidos de forma aleatoria por todo el escenario y tendrán el resto de variables con valor 0, debido a que byExpert, METHODKNOWLEDGED y STATE cambiarán su valor según la situación en la que el agente conocerá el incedio y KNOWDISASTER es la que inicia el comportamiento de después de conocer la situación del desastre, por lo tanto es conveniente que esté a 0 hasta que los agentes tengan el conocimiento. En el caso de DISASTER estará a 1 en los objetos que forman el incendio (llamas y humo), todos los demás objetos no tendrán este metadato.

KNOWPLACE	***EXPERIENCE***	***Número de agentes***	***%***
0	***0***	***125***	***25***
0	***1***	***125***	***25***
1	***0***	***125***	***25***
1	***1***	***125***	***25***

Tabla 4. Cantidades de agentes según KNOWPLACE y EXPERIENCE.

Las cantidades totales de agentes de cada tipo se han distribuido de esta manera debido a la escasez de información verídica sobre las cantidades y características de personas antes del desastre.

Con esta simulación se pretende observar, una multitud casual con comportamientos tanto individuales como colectivos en el marco del algoritmo propuesto,

una expansión de la información sobre la existencia del desastre o norma emergente en la que muy pocos agentes se enteren de forma directa y unos movimientos diferentes y aleatorios de, al menos, los agentes que no han conocido el desastre. Los objetivos finales son, el total de seguidores sea mayor que el total de fugados entre los agentes inexpertos, los agentes inexpertos sean el menor número posible y en caso de haber, estén el menor tiempo posible en estado de pánico.

5.1. Resultados y comparación con estudios sociológicos.

En la figura 26 se puede apreciar la simulación en el paso 30. Los pasos en la simulación es la variación de al menos un agente en su posición X,Y en el entorno, esto hace que se produzca una “foto” diferente de todo el entorno cada vez que pulsa el botón play. Se observa en la figura como hay solo 16 agentes que se han dado cuenta de que existe el incendio (representado por un rectángulo marrón, un poco desplazado a la derecha del centro de la foto) y todos están relativamente cerca de él, por lo que el intercambio de información sobre él (la norma) o las normas aún están poco expandidas. Pueden apreciarse ya todos los comportamientos posibles, desde un agente inexperto paralizado (representado con un triángulo rojo justo al lado izquierdo, a la misma altura que el incendio) hasta agentes expertos con seguidores (expertos representados por triángulos azules y seguidores representados por triángulos verdes) situados por encima y a la izquierda del incendio.

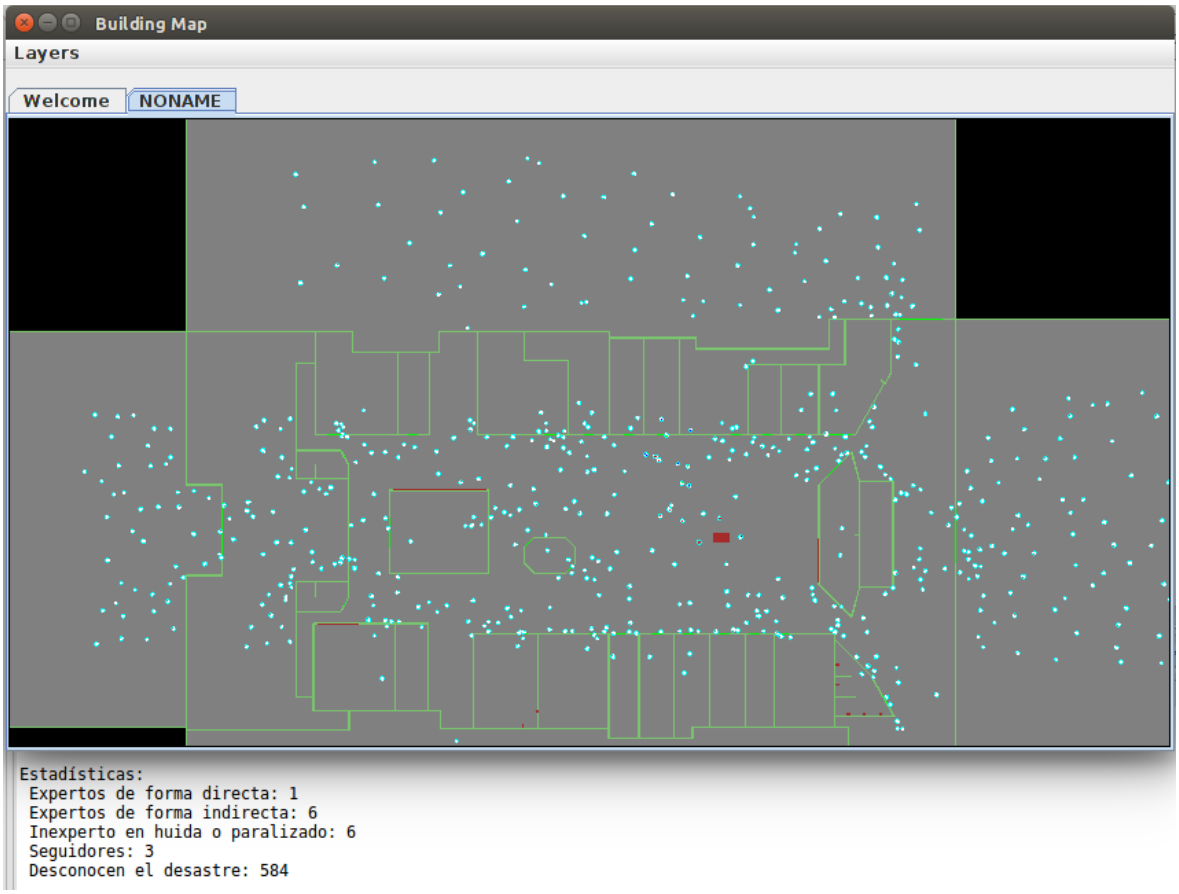


Figura 26. Paso 30 de unas de las simulaciones. La habitación inferior que actua como calle está cortada en esta imagen debido a la necesidad de ampliar la imagen para apreciar colores y agentes.

En la figura 27, en el paso 160 de la simulación, permanecen 60 agentes sin conocimiento sobre la existencia del desastre, se puede destacar que hay agentes por fuera del centro comercial que sin haber entrado a éste conocen que existe un incendio (en el interior círculos negros en la imagen) y algunos agentes, estando dentro del centro comercial no tienen conocimiento de que existe un incendio aún (seleccionados con círculos amarillos). Hay un intercambio de información o unas normas expandidas de forma avanzada. En los totales se puede ver como en el caso de los agentes inexpertos, son mayoría los que actúan de forma cívica (seguidores representados en triángulos verdes) y varios de los que no ya han huido zonas seguras (seleccionados con círculos rojos).

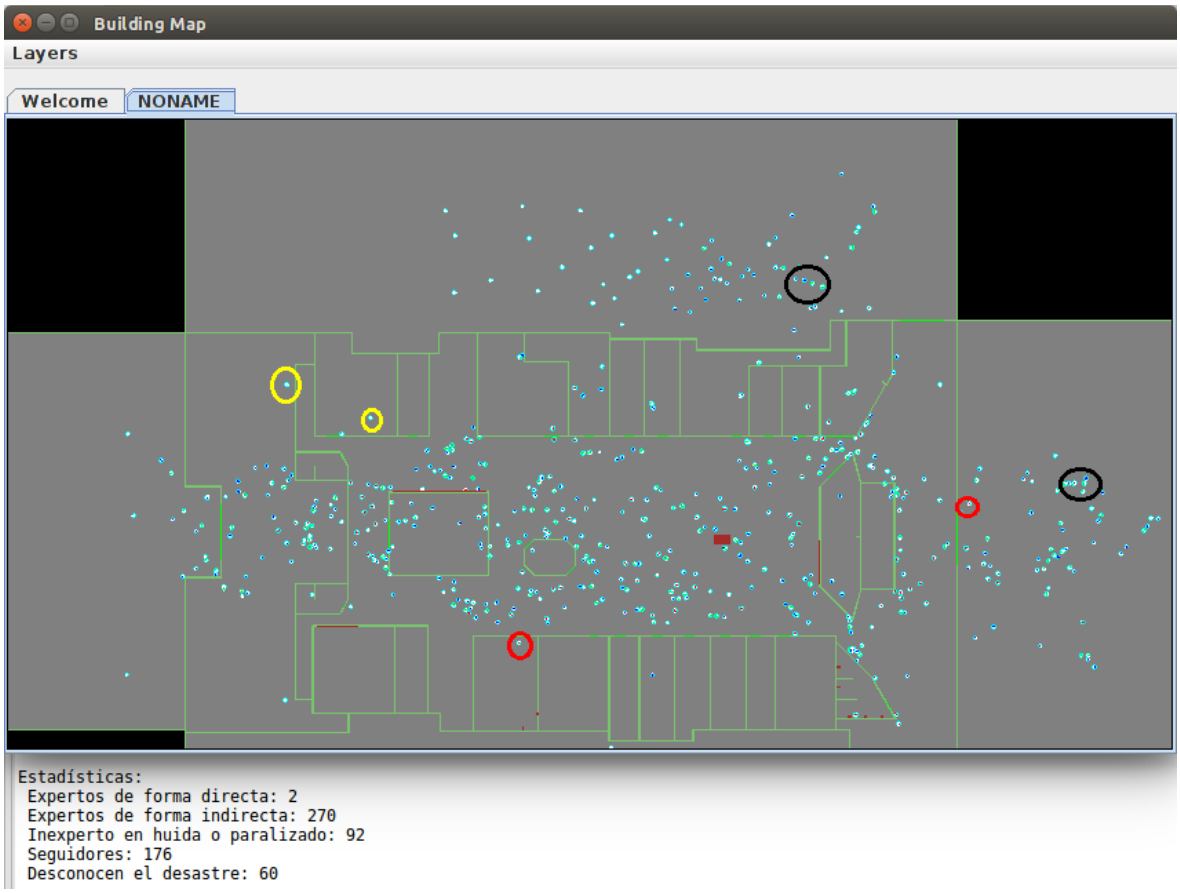


Figura 27. Paso 160 de una de las simulaciones. La habitación inferior que actua como calle está cortada en esta imagen debido a la necesidad de ampliar la imagen para apreciar colores y agentes.

En la figura 28, en el paso 482, todos los agentes ya tienen conocimiento de que hay un incendio, de esta forma se presentan los dos tipos de este algoritmo: agentes inexpertos y agentes expertos. Las normas expandidas o el intercambio de información sobre el incendio está totalmente propagado. Muchos de los inexpertos fugados ya han completado la acción (representados dentro de círculo rojo). Dos de los expertos tienen tres seguidores y están en acción de evacuar (representados como grupos con un triángulo azul y tres triángulos verdes que están cerca y apuntan a la misma dirección, encerrados todos en círculos verdes en la figura). Para los demás expertos no será posible la evacuación debido a que no hay posibilidad de que alcancen las cifra de tres seguidores, al no haber ni inexpertos en pánico ni inexpertos que no conozcan el incendio.

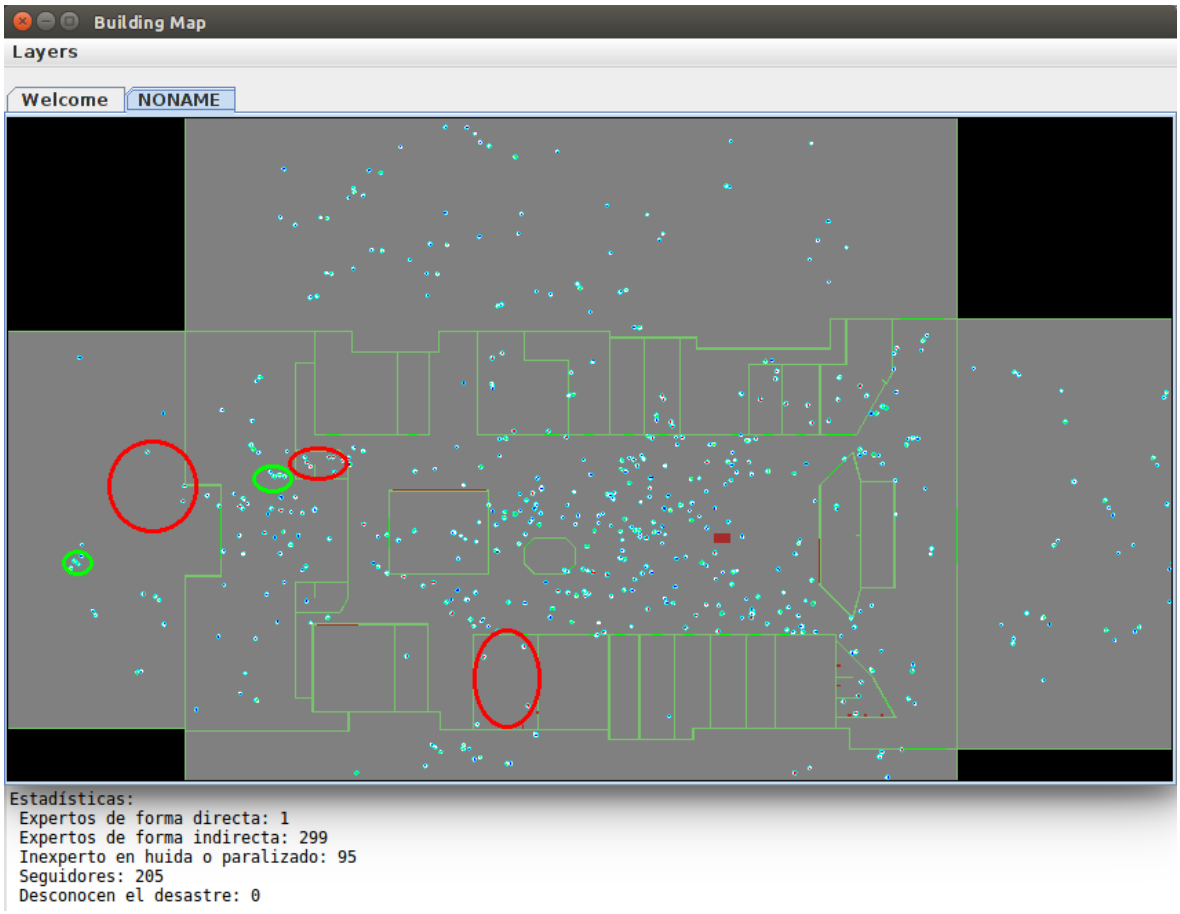


Figura 28. Paso 482 de una de las simulaciones. La habitación inferior que actua como calle está cortada en esta imagen debido a la necesidad de ampliar la imagen para apreciar colores y agentes.

En el transcurso de la simulación de las figuras 26, 27 y 28. Se observa como el agente experto cuando lleva seguidores, evita que éstos tengan conocimiento sobre el incendio de otra manera, descartando la posibilidad de que estos seguidores fueran agentes en pánico o en huida, esto coincide con la definición en el trabajo de Fidalgo [43][44] de lo que tiene que hacer un individuo con experiencia o líder. Fidalgo defiende que un líder tiene gran importancia en una emergencia y evacuación, debe cortar o desacelar el proceso de conducta desordenada y caótica. Fidalgo [43][44] hace una observación sobre el pánico en una de sus guías, defiende que casi todos los expertos coinciden en que los casos de pánico son escasos y muy localizados. En esta simulación (ilustrada por las tres figuras) solo se detecta un caso de pánico, por lo tanto coincide con la observación en la guía.

Después de 1500 simulaciones con idénticas condiciones de inicio a la ilustrada en

las figuras 26, 27 y 28. Los resultados (medias globales) de cambios de comportamiento son los registrados en la tabla 5. Estos datos se han obtenido con valores en las simulaciones como son: simulación más larga registrada de 798 pasos para que todos los agentes tengan conocimiento del desastre, simulación más corta registrada de 270 pasos en la misma situación y valor medio de pasos 532,5947 con idénticas características. En más de un 90% de las simulaciones, el último agente en conocer el incendio se encontraba fuera del recinto (en la habitaciones que actúan de calles).

Comportamiento	***Valor máximo***	***Valor mínimo***	***Valor medio***	***%***
Expertos conocen el desastre de forma directa	***3***	***0***	***1,5487***	***0,2581***
Expertos conocen el desastre de forma indirecta	***300***	***297***	***298,4513***	***49,7419***
Inexpertos que huye o están paralizados	***129***	***91***	***109,7553***	***18,2926***
Seguidores	***209***	***171***	***190,2447***	***31,7074***

Tabla 5. Resulados de cambios de comportamientos en 1500 simulaciones.

De estas simulaciones surgieron también otros resultados de importancia, como son los agentes que en algún momento entraron en pánico durante la emergencia, los agentes que lograron salir (antes de la condición general de salida que está programada para que empiece a ejecutarse por todos los agentes en el “paso” 1000 de las simulaciones) y en cuantos pasos, cuantos pasos dieron los agentes despues de la condición de salida general.

Resultados	*Valor máximo*	*Valor mínimo*	*Valor medio*	*%*
Pánico en agentes	*3*	*0*	*1,512*	*0,252*
Agentes que salieron antes de los 1000 pasos	*122*	*88*	*104,7093*	*17,4516*
Pasos totales de los agentes que salieron antes de los 1000 pasos	*917*	*515*	*716,492*	*X**
Pasos totales de los agentes que salieron después de los 1000 pasos	*1673*	*1902*	*1787,492*	*X**

**** La X que no hay una referencia para hallar ese parámetro.***

Tabla 6. Resulados de cambios de comportamientos en 1500 simulaciones.

Al clasificar y agrupar los datos de la tabla 5 y 6, se obiene que el 81,7074% (suma de los dos tipos de expertos y seguidores) se comportaron de manera calmada, por otro lado el 18,2926% fueron responsables de una conducta no cívica, de éstos, el 0,252% fue presa del pánico. Fidalgo [43][44] afirma que las reacciones de confusión, ansiedad, paralización, gritos histéricos y pánico, son del 10-25% en una catástrofe, entre este rango se encuentra el número de agentes inexpertos que tuvieron una actitud no cívica en los resultados finales de las 1500 simulaciones (medias globales). También concuerda el 0,252% de agentes que fue presa de pánico con la observación, anteriormente mencionada de Fidalgo [43][44], que defendía que todos los expertos coinciden en que los casos de pánico son escasos y muy localizados.

6. Conclusión y trabajo futuro.

6.1. Conclusión.

Se ha propuesto una formalización del comportamiento de usuarios según el modelo de Fidalgo y de la norma emergente de Turner y Killian, con este algoritmo se ha intentado abarcar varias reacciones y comportamientos en este trabajo, que han sido documentados por diversos autores en estudios sociológicos anteriores sobre estos fenómenos, con la dificultad de contar con solo una persona, novata en el tema, para el desarrollo y la implementación. Con ayuda de la tutorización y la herramienta MASSIS, los comportamientos implementados se asemejan a los redactados en las guías de buenas prácticas de Manuel Fidalgo. Se ha logrado representar comportamientos individuales como el pánico en agentes inexpertos o la fuga, pero también se ha logrado con la base de la teoría de la norma emergente de Killian y Turner que surjan intereses comunes entre los individuos, como estar a salvo del desastre, posibilitando la aparición de comportamientos colectivos para la evacuación. En cuanto al entorno, se ha conseguido representar una gran superficie a escala, cierto es que en el caso de añadir plantas, el escenario y la posible evacuación cambiarían totalmente. En líneas generales queda mucho por añadir y mejorar, este trabajo ha tenido como resultado, una aproximación para intentar incidir en la importancia sobre todo en variables como la experiencia o conocimiento en evacuación y desastres de los individuos, más allá de predecir y comprender el comportamiento humano en desastres. Quizás en un futuro, con estudios en psicología social o sociología que puedan descomponer mejor y explicar las razones junto con los sentimientos de las personas y categorizar sin error a cada una, se llegará a predecir el comportamiento humano con un porcentaje alto de acierto puesto que las herramientas y el cálculo computacional, en un futuro cercano o incluso hoy en día, es capaz de dar grandes soluciones.

6.2. Trabajo futuro.

Este algoritmo es solo un acercamiento para lo que en realidad es posible hacer en este tema y con esta herramienta. Los puntos a mejorar en este trabajo son los siguientes.

Adición de más factores y comportamientos. Con esto se dotará al algoritmo de mayor capacidad para estudiar y verificar comportamientos como causas y así poder extraer resultados mas cerca de la realidad.

Entornos más grandes o edificios con varias plantas. Con una etapa de desarrollo mayor será posible ampliar el entorno, por ejemplo al rango de una ciudad a escala con pisos de varias plantas, que introducirán al algoritmo en otro contexto diferente.

Perfeccionamiento de los movimientos. Esta versión contiene movimientos que no están tan cerca de los que producen los individuos en la realidad como los que se podrían implementar. Un ejemplo de ello es el movimento aleatorio de los individuos antes del conocimiento del incendio que dista del comportamiento real de las personas en la visita a un centro comercial.

Situación, mobiliario y agentes diferentes. Solo se ha simulado con una situación, tipo de desastre y de población, se podrían mejorar los resultados con simulaciones en varias situaciones como varios focos de incendios, diferentes posiciones de mobiliario, difentes proporciones de cada tipo de agentes, entre otras posibilidades.

7. Conclusion and future work.

7.1. Conclusion.

It has been proposed an implementation of the behavior ussing MASSIS an according to the Fidalgo model and the Emergent norm theory of Turner and Killian. Using this implementation several reactions and behaviors in this work has been considered. These reactions have been studied as well in sociological studies and the results of the implementation briefly compared to them. The results show similarities to those written in good practices guides of Manuel Fidalgo. Individual behaviors such as panic or escaping have been achieved. It has also been established, using Killian and Turner's emerging norm theory, that common interests arise among individuals, such as being safe from disaster, making possible the emergence of collective behaviors for evacuation. As for the environment, it has managed to represent a large area through the incorporation of floors. In this area, the scenario and possible evacuation change completely when actors know placer or not. Despite the possible improvements, this work has shown the importance of variables such as experience or knowledge in evacuation and disasters of individuals, whitout the need of understanding the human behavior in disasters. Perhaps in the future, with more accurate studies in social psychology or sociology to better decompose and explain the compelling reasons that govern people's behavior, we could predict human behavior with a high percentage of success.

7.2. Future work.

This algorithm is only an approach for what is actually possible in this topic and with this tool. The points to improve this work are the following.

Addition of more factors and behaviors. With this, the algorithm has greater capacity to study and verify behaviors and thus to be able to extract results closer to reality.

Larger environments or buildings with several floors. With a greater developmental stage it will be possible to extend the environment, for example to the rank of a city on a scale with edificios of several floors, that introduces the algorithm in another different context.

Improvement of movements. This version contains movements that are not as close to those produced by individuals in reality as those that could be implemented. An example of this is the random movement of individuals before the fire knowledge that is far from the actual behavior of the people in the visit to a shopping center.

Situation, furniture and different agents. Only have a simulation with a situation, type of disaster and population, results can improve with simulations in various situations such as several fires, different positions of furniture, different proportions of each type of agents, among other possibilities

A. Apéndice.

La clase ObjectHighLevelController, como se puede ver en la figura A.1, se compone de los siguientes elementos:

- El constructor de la clase requiere tres parámetros:

  - Una referencia de LowLevelAgent es agent. Los agentes en MASSIS tienen dos componentes: uno controla la decisiones del agente (HighLevelController) y otro se encarga de percibir el entorno (LowLevelAgent). LowLevelAgent transfiere a HighLevelController la información necesaria para tomar decisiones. HighLevelController cambia a LowLevelAgent, hace que se mueva, cambie sus propiedades, entre otras acciones.

  - Metadata es un mapa que almacena todas las entradas valor-clave (metadatos) que se introducen en el agente a través del editor de entorno.

  - Resourcesfolder es la carpeta donde el agente puede encontrar algunos archivos o dispositivos para realizar operaciones de E/S. Para el algoritmo de comportamiento de multitudes en desastres no será necesaria información de una carpeta externa.

  Dentro del constructor se invoca a la superclase con la palabra clave super y los tres parámetros antes mencionados. Con esto, se busca almacenar estos parámetros en la superclase. Como también se almacena el contenido AgentHighLevelController en el propio objeto agente, mediante el método setHighLevelData. Con esto último, se consigue acceder a los metadatos asociados con el método getProperty de la clase agente.

  Se recoge a través de metadata si existe la entrada DISASTER en los

metadatos del objeto (agente) y se almacena en disasterStr.

- IsDisaster. Es un método de tipo boolean que devuelve si el objeto (agente) tiene la propiedad de ser un desastre o no. Utiliza el método getProperty del objeto agent.

- SetDisaster(boolean disaster). Utiliza el método setProperty del objeto agent, en el cual envía los parámetros DISASTER para elegir esa entrada y el valor al cual se quiere cambiar la entrada (el valor del parámetro recibido disaster).

- Stop. Antes de que se termine la simulación se llama a este método. Se puede utilizar para liberar memoria y terminar.

- Step. Es donde ocurre la lógica del objeto (agente), en cada avance de la simulación se provoca una llamada a este método.

  En este caso no ocurre nada debido a que no se busca recrear ningún comportamiento, solo se quiere que el objeto tenga propiedad de desastre.

```
public class ObjectHighLevelController extends HighLevelController {

    private static final long serialVersionUID = 1L;

    public ObjectHighLevelController(LowLevelAgent agent, Map<String, String> metadata, String resourcesFolder) {
        super(agent, metadata, resourcesFolder);
        this.agent.setHighLevelData(this);

        String disasterStr = metadata.get("DISASTER");

        if (disasterStr == null || !"true".equals(metadata.get("DISASTER"))) {
            this.setDisaster(false);
        } else {
            this.setDisaster(true);
        }

    }

    public boolean isDisaster() {
        return "true".equals(this.agent.getProperty("DISASTER"));
    }
    public void setDisaster(boolean disaster) {
        this.agent.setProperty("DISASTER", String.valueOf(disaster));
    }

    @Override
    public void stop() {
        /*
         * Clean resources, threads...etc
         */
    }

    @Override
    public void step() {

    }
}
```

Figura A.1. Código de la clase ObjectHighLevelController.

La clase AgentHighLevelController contiene una estructura similar a la de la clase ObjectHighLevelController (las dos heredan de la clase HighLevelController) pero ampliada y modificada en algunos métodos. Esto es así porque en esta clase, se ha buscado representar todos los comportamientos de los agentes. Por lo tanto, todas las propiedades del agente (KNOWDISASTER, KNOWPLACE, METHODKNOWLEDGED, EXPERIENCE y STATE), tienen su método para establecer el valor true o false y su método booleano que devuelve el estado de la propiedad en el objeto agent. De la misma manera, en el constructor, se almacena en disasterStr si no existe o el valor de cada propiedad del agente en metadata, y gracias a los mecanismos antes comentados, es posible que haya variaciones en estos valores durante la simulación. Cabe mencionar que además de estas semejanzas anteriormente expuestas, la versión y los métodos stop son idénticos en ambas clases.

```
public AgentHighLevelController(LowLevelAgent agent, Map<String, String> metadata, String resourcesFolder) {
    super(agent, metadata, resourcesFolder);
    this.agent.setHighLevelData(this);

    String disasterStr = metadata.get("KNOWPLACE");

    if (disasterStr == null || !"true".equals(metadata.get("KNOWPLACE"))) {
        this.setKnowPlace(false);
    } else {
        this.setKnowPlace(true);
    }

    disasterStr = metadata.get("EXPERIENCE");

    if (disasterStr == null || !"true".equals(metadata.get("EXPERIENCE"))) {
        this.setExperienced(false);
    } else {
        this.setExperienced(true);
    }
```

Figura A.2. Parte del constructor de la clase AgentHighLevelController.

Una vez expuestos los elementos idénticos o ampliados entre la clase AgentHighLevelController y ObjectHighLevelController. Se procede a explicar los demás elementos de las clase AgentHighLevelController.

Algunas variables y una constante para facilitar la ejecución de procesos:

- Search_range. Es una constante de tipo double. Con valor 300 (cm) se usa para establecer el rango de distancia para percibir otros agentes o el desastre.

- CurrentTarget. Contiene un objeto de tipo Location. La clase Location se usa principalmete para guardar las coordenadas de una posición y establecerla como objetivo al cual se tiene que mover un agente. También se puede usar para obtener

el objeto Floor y de éste, obtener habitaciones, obstáculos, agentes, etc. Se usa en el método MoveRandomly.

- ByExpert.Variable de tipo integer. Puede tomar los valores 0, 1 o 2. se utiliza en el caso de que el agente se entere por otro agente del desastre. Tiene valor 0 cuando el agente no se ha enterado o se entera de manera directa del desastre, 1 si el agente se entera por otro agente inexperto y 2 si el agente se entera por un agente experto.

- DisasterLocation. De tipo Location. Se utiliza en los agentes inexpertos, que conocen de forma indirecta por otro inexperto o de forma directa (excepto los que no conocen el lugar), para almacenar la posición del desastre. Es parte del proceso de ir a ver qué pasa de un agente experto que conoce el desastre por un agente inexperto que está huyendo.

- NumberOfFollowers. Es de tipo integer. Contiene el número de seguidores inexpertos que siguen a un agente.

- Stop. Variable boolean. Se ha creado con la finalidad de detener el movimiento en la evacuación o huida de los agentes, cuando éstos ya han llegado al primer punto objetivo de la zona o habitación a la que debían ir. De este modo se evita que una vez hayan llegado a la zona segura o más alejada del desastre siga moviéndose.

- Destination. De tipo Location. Se usa para almacenar la posición a la que debe ir un agente en los métodos: escapeToFarthestZone (escapar a la zona mas alejada del desastre) y escapeToSafeZone (escapa a una zona segura).

- Randomtarget. Contenedor de tipo Location. Se utiliza en el método followMe (método sígueme, usado por agentes con experiencia) para almacenar la posición a la que debe ir el agente.

- FollowTarget. De tipo LowLevelAgent. Se usa en agentes seguidores y contiene la referencia del agente al que siguen. Elemento importante en el método iAmFollowing.

- Count. Variable de tipo integer. Almacena el número de veces que es ejecutado el método step. Se utiliza para conseguir que los agentes expertos que no consigan llegar al número de seguidores necesarios, evacuen a los que llevan o salgan solos hacia la zona mas alejada del desastre o a una zona segura.

```
private static final long serialVersionUID = 1L;
private static final double search_range = 300;
private Location currentTarget;
private int byExpert=0;
private Location disasterLocation = null;
public int numberOfFollowers = 0;
private boolean stop = false;
private Location destination = null;
private Location randomTarget = null;
private LowLevelAgent followTarget = null;
private int count = 0;
```

Figura A.3. Variables globales y constantes de AgentHighLevelController.

A.1. Métodos de AgentHighLevelController sin capacidad de mover al agente.

Step. Método de tipo void. Cada vez que hay un avance en la simulación se le llama. Se encarga de ejecutar el método moveRandomly cuando el agente aún no conoce el desastre. Si ya conoce el desastre, ejecuta ExpertBehaviour si el agente es experto, o InexpertBehaviour en caso contrario. Para no dejar a expertos, que no llegan al número de seguidores para evacuar, moviéndose aleatoriamente, se lleva la cuenta de las llamadas que se han realizado al método durante la ejecución. Con esta cuenta, almacenada en la variable count, se consigue que los expertos a partir de un número de pasos en la simulación (en el caso de la figura A.4 es 1000), evacuen a los seguidores que guían o salgan ellos solos hacia una zona segura (en caso de que no conozcan el entorno), ejecutando escapeToSafeZone, o a la zona más alejada del desastre (en caso de que conozcan el entorno), ejecutando escapeToFarthestZone.

```java
@Override
public void step() {

    if(getCount() < 1000 || !this.isExperienced()) {
        if(!stop){
            if(!this.isKnowDisaster()){
                moveRandomly();
            }
            else {
                if (this.isExperienced()) {
                    ExpertBehaviour();
                } else {
                    InexpertBehaviour();
                }
            }
        }
    }
    else
        if(this.isExperienced() && this.isKnowPlace() && !stop)
            escapeToFarthestZone();
        else if(this.isExperienced() && !this.isKnowPlace() && !stop)
            escapeToSafeZone();

    setCount(getCount()+1);
}
```

Figura A.4. Método step de la clase AgentHighLevelController.

InexpertBehaviour. Método de tipo void. Se ocupa de adjudicar un comportamiento a los agentes inexpertos después de que hayan conocido el desastre. Para ello, hace una criba en primer lugar en función del valor de la propiedad KNOWPLACE, luego diferencia a los agentes que tengan METHODKNOWLEDGED a true de los que lo tengan a false. En el caso de los agentes con esta última propiedad a true, se diferencia entre los que conocieron el desastre por un experto, byExpert con valor 2, de los que conocieron el desastre por otro agente inexperto, byExpert con valor 1. La forma de asignar los comportamientos después de diferenciar a cada agente es la de tabla de la sección 3.1.1. Relaciones entre variables y comportamientos.

```
void InexpertBehaviour(){
        if(isKnowPlace()){
            if(isMethodKnowledged()){
                if(isbyExpert() == 2){
                    iAmFollowing(isFollowingSomeone());
                    followExpert(search_range);

                }
                else if(isbyExpert() == 1){
                    setState(false);
                    escapeToFarthestZone();
                }
            }
            else{
                setState(false);
                escapeToFarthestZone();
            }
        }
        else {
            if(isMethodKnowledged()){
                if(isbyExpert() == 2){
                    iAmFollowing(isFollowingSomeone());
                    followExpert(search_range);
                }
                else if(isbyExpert() == 1){
                    setState(false);
                    escapeToSafeZone();
                }

            }
            else{
                followExpert(search_range);
                iAmFollowing(isFollowingSomeone());
            }
        }
}
```

Figura A.5. Método InexpertBehaviour de la clase AgentHighLevelController.

ExpertBehaviour. Método de tipo void. Se encarga de asignar comportamiento a los agentes expertos después de haber conocido el desastre. Selecciona en primer lugar en teniendo en cuenta el valor de la propiedad METHODKNOWLEDGED, luego en función de como esté KNOWPLACE. En un paso intermedio entre revisar estas dos propiedades comentadas, en el caso de los agentes con METHODKNOWLEDGED con valor true, se distigue entre los que tuvieron información del desastre por un experto, byExpert con valor 2, de los que la tuvieron por otro agente inexperto, byExpert con valor 1. La forma de asignar los comportamientos después de diferenciar a cada agente es la de tabla de la sección 3.1.1. Relaciones entre variables y comportamientos.

```
void ExpertBehaviour(){
        if(isMethodKnowledged()){
            setState(true);
            if(isbyExpert() == 2){
                if(isKnowPlace()){
                    followMe();
                    if(numberOfFollowers == 3) {
                        escapeToFarthestZone();
                    }
                }
                else{
                    followMe();
                    if(numberOfFollowers == 3) {
                        escapeToSafeZone();
                    }
                }
            }
            else if (isbyExpert() == 1){
                if (getDisasterLocation() != null){
                    whatisUp(getDisasterLocation());
                }
                else {
                    if(isKnowPlace()){
                        followMe();
                        if(numberOfFollowers == 3) {
                            escapeToFarthestZone();
                        }
                    }
                    else {
                        followMe();
                        if(numberOfFollowers == 3) {
                            escapeToSafeZone();
                        }
                    }
                }
            }
        }
        else {
            if(isbyExpert() == 0){
                setState(false);
                if(isKnowPlace()){
                    followMe();
                    if(numberOfFollowers == 3) {
                        escapeToFarthestZone();
                    }
                }
                else{
                    followMe();
                    if(numberOfFollowers == 3) {
                        escapeToSafeZone();
                    }
                }
            }
        }
}
```

Figura A.6. Método ExpertBehaviour de la clase AgentHighLevelController.

DisasterRange(double range). Método de tipo boolean. Encuentra las entidades que están en un rango (pasado por parámetros cuando se invoca) con el método getAgentsInRange de la clase LowLevelAgent y un bucle, si alguna de estas entidades tiene activada la propiedad DISASTER, este método devuelve true. En caso de que este método sea ejecutado en un agente inexperto, cuando éste encuentra el desastre, se almacena la posición del desastre en DisasterLocation.

```
boolean DisasterRange(double range) {
    boolean disaster=false;

    for (LowLevelAgent otherAgent : this.agent.getAgentsInRange(range)) {
        if ("true".equals(otherAgent.getProperty("DISASTER"))){
            disaster = true;
            if ("false".equals(this.agent.getProperty("EXPERIENCE"))){
                setDisasterLocation(otherAgent.getLocation());
            }
        }
    }
    return disaster;
}
```

Figura A.7. Método DisasterRange de la clase AgentHighLevelController.

ByExpertRange(double range). Método de tipo integer. Escoge a los agentes en un rango, si no tienen la propiedad KNOWDISASTER activada se dejará a 0 la variable byExpert. En caso contrario, se diferencia si los agentes hallados son expertos o seguidores, si pertenecen a alguno de estos dos grupos la variable byExpert tendrá valor 2, en el caso de que los agentes sean inexpertos que huyen, byExpert pasa a tener valor 1 y se copia el valor de DisasterLocation del agente que huye a la del agente que ejecuta este método.

```
private int ByExpertRange(double range) {

    for (LowLevelAgent otherAgent : this.agent.getAgentsInRange(range)) {
        if ("true".equals(otherAgent.getProperty("KNOWDISASTER")) ){
            if ("true".equals(otherAgent.getProperty("EXPERIENCE")) || "true".equals(otherAgent.getProperty("STATE"))){
                setbyExpert(2);
            }
            else {
                setbyExpert(1);
                final AgentHighLevelController agent = (AgentHighLevelController) otherAgent.getHighLevelData();
                setDisasterLocation(agent.getDisasterLocation());
            }
        }
    }
    return isbyExpert();
}
```

Figura A.8. Método ByExpertRange de la clase AgentHighLevelController.

followMeInexpert(double range). Método de tipo void. Recoge a los agentes que están en un rango. En primer lugar, comprueba que numberOfFollowers del agente que ejecuta este método sea menor que 4 y verifica que la entidades que detectan sean agentes por medio de sus IDs. Continua contrastando que los agentes encontrados sean inexpertos, que no sigan a nadie, y que sepan donde se encuentra el desastre, o lo que es lo mismo, que EXPERIENCE tenga valor true, que FollowTarget se ajuste a null y DisasterLocation tenga

un valor diferente a null. En el caso de que se cumplan todos estos requisitos detallados, se modificarán los valores de los agentes, KNOWDISASTER pasará a tener valor true, FollowTarget almacenará una referencia del agente ejecutante y DisasterLocation pasará a estar a null. En el agente ejecutante la variable numberOfFollowers aunmentará en 1 unidad su valor.

```
private void followMeInexpert(double range){
    for (LowLevelAgent otherAgent : this.agent.getAgentsInRange(range)) {
        if (numberOfFollowers < 4 && (otherAgent.getID() < 1148 || otherAgent.getID() > 1153 ) ){
            final AgentHighLevelController follower = (AgentHighLevelController) otherAgent.getHighLevelData();
            if ("false".equals(otherAgent.getProperty("EXPERIENCE"))
            && follower.isFollowingSomeone() == null && follower.getDisasterLocation() != null){
                follower.setKnowDisaster(true);
                follower.setFollowTarget(this.agent);
                follower.setDisasterLocation(null);
                numberOfFollowers++;
            }
        }
    }
}
```

Figura A.9. Método followMeInexpert de la clase AgentHighLevelController.

followExpert(double range). Método de tipo boleean. Encuentra a los agentes dentro de un rango especificado. Examina si las entidades encontradas son agentes por medio de sus IDs y, si éstos son expertos que conocen el desastre, no se encuentran en el proceso de ir a ver qué pasa y tienen menos de 4 seguidores. Estas condiciones en términos de variables y valores son KNOWDISASTER con valor true, numberOfFollowers menor de 4, EXPERIENCE se ajusta a true y DisasterLocation a null. Para el agente ejecutante también se coteja que no siga a nadie, FollowTarget tiene null almacenado. Una vez los agentes cumplan estas condiciones, la variable FollowTarget del agente ejecutante almacenará una referencia al primer agente encontrado y la propiedad STATE de este mismo agente tendrá valor true. El número de seguidores, numberOfFollowers, del primer agente encontrado incrementará en 1 unidad. Por último el método devolverá true, interrumpiendo la ejecución del bucle, desde que se encuentre a un solo agente que cumpla todas las condiciones. En todos los demás casos, el método devuelve false.

```
private boolean followExpert(double range){
    for (LowLevelAgent otherAgent : this.agent.getAgentsInRange(range)) {
        if (otherAgent.getID() < 1148 || otherAgent.getID() > 1153 ){
            final AgentHighLevelController leader = (AgentHighLevelController) otherAgent.getHighLevelData();
            if ("true".equals(otherAgent.getProperty("KNOWDISASTER")) && leader.numberOfFollowers < 4){
                if ("true".equals(otherAgent.getProperty("EXPERIENCE"))
                && this.isFollowingSomeone() == null && leader.getDisasterLocation() == null){
                    setFollowTarget(leader.agent);
                    leader.numberOfFollowers++;
                    setState(true);
                    return true;
                }
            }
        }
    }

    return false;
}
```

Figura A.10. Método followExpert de la clase AgentHighLevelController.

A.2. Métodos de AgentHighLevelController con capacidad de mover al agente.

Para poder tener movimiento, los agentes necesitan información sobre el entorno, como la habitación en la que se encuentran. Este conocimiento es transmitido por la clase LowLevelAgent que proporciona los métodos necesarios.

La estructura en la que se han basado todos los métodos con capacidad de mover al agente es la que se puede observar en la figura A.11. En ella se distinguen los siguientes elementos:

- La obtención de la habitación en la que está el agente. Las habitaciones están representadas por la clase SimRoom. La obtención de la habitación del agente se hace llamando al método getRoom.

- Posición a la que se moverá el agente, que en este caso es a un lugar aleatorio en la habitación actual. Las posiciones están diseñadas por la clase Location. Para lleva a cabo este proceso se define una variable de tipo Location que se llama currentTarget y se almacena en ella, una posición aleatoria de la habitación

actual mediante el método getRandomLoc.

- Hacer que el agente se mueva al punto almacenado. Para conseguir esto, es necesario usar el método approachTo. Este método trata de acercar al agente a la posición asignada, evitando obstáculos. ApproachTo necesita que se le envíen dos parámetros, La posición a la que se quiere mover al agente (en este caso CurrentTarget) y el método ApproachCallback.

El método ApproachCallback será llamado cuando el proceso que calcula la ruta (pathfinder) se haya terminado en el paso actual. Consta de 3 posibilidades: que el agente llegue a la posición deseada, onTargetReached, que se aproxime un paso a la posición deseada, onSuccess o que suceda un error de algun tipo, onPathFinderError. En el ejemplo de la figura A.12, el agente solo se mueve hasta llegar a la posición almacenada en currentTarget y permanece ahí, esto es debido a que no se le da valor null a currentTarget en el método onTargetReached, con esto, se asignaría otra posición a alcanzar por el agente. Ejecutado con éxito este método, ya se puede hacer llamar al método approachTo en el LowLevelAgent.

```
if (this.currentTarget == null) {
     SimRoom currentRoom = this.agent.getRoom();
     this.currentTarget = currentRoom.getRandomLoc();
}

ApproachCallback callback = new ApproachCallback() {
    @Override
    public void onTargetReached(LowLevelAgent agent) {

    }

    @Override
    public void onSucess(LowLevelAgent agent) {

    }

    @Override
    public void onPathFinderError(PathFinderErrorReason reason) {
        // Error!
        Logger.getLogger(MyHelloHighLevelController.class.getName())
           .log(Level.SEVERE,
           "Error when approaching to {0} Reason: {1}",
           new Object[] { currentTarget, reason });
    }
};

this.agent.approachTo(this.currentTarget, callback);
```

Figura A.11. Modelo de estructura de los métodos con capacidad de mover al agente en AgentHighLevelController.

MoveRandomly. Método de tipo void. En esta ocasión, la selección de la posición a la que se va a mover al agente es mas compleja que en la figura A.11. Para ello, primero se obtiene la posición del agente con el método getLocation, luego se accede al suelo mediante el método getFloor. Luego se accede a una habitación aleatoria, para este paso previamente se definió una lista auxiliar para almacenar la lista de habitaciones, y un número al azar entre 0 y el número de habitaciones con la clase importada Random. Como último paso para guardar el objetivo en CurrentTarget, se accede a una posición al azar dentro de la habitación aleatoria elegida anteriormente.

En el método callback, a cada paso, en OnSucess, se comprueba si en el rango del agente se encuentra el desastre o si hay otro agente que conozca el desastre. Esto se hace examinando los valores recibidos por DisasterRange y ByExpertRange respectivamente. En el caso de que se cumpla la primera condición, KNOWDISASTER tendrá valor true y METHODKNOWLEDGED valdrá false, mientras que si se cumple la segunda condición, tanto KNOWDISASTER y METHODKNOWLEDGED se ajustarán al mismo valor, a true. En el caso de que el agente llegue a la posición objetivo y no cumpla ninguna condición, la posición objetivo o currentTarget pasará a ser null y se le asignará una nueva.

FollowMe. Método de tipo void. Es muy parecido a MoveRandomly, en este caso la posición objetivo se almacena en la variable randomTarget pero la forma de elegirla entre todas las habitaciones, al azar, es idéntica. Otra diferencia es la llamada en OnSucces, del método Callback, a followMeInexpert con un rango. La característica mas relevante que también diferencia este método respecto del anterior, es que desde el comienzo está sometido a un condicional que se asegura que, el número de seguidores del agente ejecutante es menos de 4, numberOfFollowers menor de 4. Con esto se cumple que el agente no ejecute este método y siga con su comportamiento utilizando los métodos de huida/evacuación.

```
private void followMe(){
    if(numberOfFollowers < 4){
        if (this.randomTarget == null) {
            Random rnd = ThreadLocalRandom.current();
            Location agentLocation = agent.getLocation();

            @Override
            public void onSucess(LowLevelAgent agent) {
                followMeInexpert(search_range);
            }
```

Figura A.12. (**Arriba**) Condición numberOfFollowers menor que 4 y uso de randomTarget en el método followMe de la clase AgentHighLevelController. (**Abajo**) método onSuccess, perteneciente a Callback del método followMe, en el que se llama al método followMeInexpert pasándole el parámetro search_range.

IAmFollowing. Método de tipo void. Se le pasa por parámetros la referencia a un agente y la ubicación en cada momento de esta referencia será la posición objetivo. Todo el método está sujeto a la condición de que la referencia no sea nula, agent diferente de null. Si se cumple esto, la propiedad STATE pasará a valer true y se aproximará el agente ejecutante al agente objetivo un paso. Cabe destacar, que además de lo anterior, este método se distingue por definir ApproachCallback en la llamada a approachTo, concretamente dentro de los parámetros.

WhatisUp. Método de tipo void. Se le pasa como parámetro la posición de desastre para llegar hasta él. A cada paso en el método OnSucces de ApproachCallback, se verifica si el desastre se encuentra en el rango de visión, esto se lleva a cabo mediante un condicional que, comprueba el valor que retorna el método DisasterRange con parámetro search_range. En caso de que se encuentre el desastre en el rango del agente, DisasterRange devuelve valor true, por lo que el agente modifica el valor de DisasterLocation a null. Cuando se cumple esto último, el agente no vuelve a entrar en este método por un condicional en ExpertBehaviour que comprueba el valor de DisasterLocation.

EscapeToSafeZone. Método de tipo void. Aquí se busca una posición en una habitación alejada (por lo tanto segura) del desastre y se mueve el agente hasta ella. Para ello, nada más empezar se declaran dos variables de tipo integer, aux y last. En aux se almacena el número que corresponde a la mitad del total de habitaciones (si se tienen 9 habitaciones se almacena el número 4), y en last se almacena el total de habitaciones menos 1 unidad (si se tienen 9 habitaciones se almacena el número 8). Además de un objeto Random con nombre rnd y una variable para almacenar el número de habitación (dentro de la colección) en la que está la posición objetivo.

La búsqueda de una posición objetivo para desplazar al agente está dentro de un condicional que hace cumplir que, el objeto de tipo Location, destination tiene valor null. Si se cumple esto, se busca el índice que tiene la habitación actual del agente dentro de la colección de habitaciones ordenadas por distancia respecto a la habitación actual. Esta colección viene dada por el método getRoomsOrderedByDistance, que usa el algoritmo de búsqueda en anchura para ordenar la colección. Con el índice obtenido, se comprueba si éste es menor o igual al valor de aux. En caso afirmativo, se obtiene un número aleatorio desde el valor de aux con incremento de una unidad hasta el valor de last (siguiendo con el ejemplo, desde 5 a 8). En caso negativo, se halla un número aleatorio desde cero hasta el valr de aux (desde 0 hasta 4). En ambos casos, se almacena el número generado en dummy. Para finalizar, se obtiene en la colección de habitaciones ordenadas por distancia, la habitación que ocupa el número almacenado en dummy y en esta habitación se obtiene la posición a la que se tiene que desplazar el agente para ponerse a salvo. Esta posición se almacena en destination.

El agente se mueve hasta la posición y una vez llega, dentro del método onTargetReached de ApproachCallback, hace que la variable stop pase a almacenar el valor true para que no varíe la posición objetivo y también cambia DisasterLocation a null.

EscapeToFarthestZone. Método de tipo void. Prácticamente idéntico a escapeToSafeZone. En este no se declaran variable dummy, ni el objeto Random rnd. La búsqueda de la posición objetivo también es muy parecida, con diferencias en la asignación de la habitación que contiene esa posición. Estas diferencias son, se escoge directamente una posición en la habitación con puesto igual al valor de last en la colección, cuando el índice de la habitación actual del agente, variable i en el código, es menor o igual a lo que almacena aux. En el caso de no cumplirse esto último, la habitación escogida, de la colección ordenada por distancia, es la de la que tiene índice cero.

Bibliografía.

[1] George H. Mead: *Mind, Self and Society from the standpoint of a social behaviorist*. Chicago University of Chicago press (1934).

[2] William McDougall: *Psycho-Analysis and Social Psychology*. Methuen & co. Ltd. Londres (1936).

[3] Sigmund Freud: *Three Essays on the Theory of Sexuality*. Franz Deuticke. Leipzig y Viena (1905).

[4] Sergio Ramírez, Rafael Gónzalez: *Aportaciones del estudio de la conducta colectiva al conocimiento y manejo de las multitudes*. Facultad de Ciencias Políticas y Sociología. U.C.M. Madrid (2015).

[5] Luis R. Izquierdo, José M. Galán, José I. Santos, Ricardo del Olmo: *Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas*. Universidad de Burgos (2008).

[6] Martin Hilbert: *Modelo basado en agentes autónomos*. Youtube. Universidad de California en Davis (2013).

[7] Ralph Turner, Lewis M. Killian: *Collective Behavior*. Prentice Hall. Nueva Jersey (1972).

[8] Guylene Proulx: *Movement of People: The Evacuation Timing*. SFPE Handbook of Fire Protection Engineering. Bethesda, MD. Third Edition. Society of Fire Protection Engineers. EEUU (2002).

[9] Steve Gwynne, Edward R Galea, M. Owen, Peter J. Lawrence: *Escape as a Social Response*. Dowden, Hutchinson & Ross. Boston (1997).

[10] Steve Gwynne, Edward R Galea, Peter J. Lawrence: *The introduction of the social adaption within evacuation modelling*. Fire and Materials. John Wiley & Sons, Ltd. EEUU (2006).

[11] Jose Aguilar: *Sistemas Multiagentes*. Facultad de Ingeniería de la Universidad de los Andes. Venezuela (2005).

[12] Tulio José Marchetti, Alejandro Javier García: *Plataformas para Desarrollo de Sistemas Multiagente. Un análisis comparativo*. Universidad Nacional del Sur. Argentina (2003).

[13] Beatriz Cortés: *Acerca de la dinámica psicosocial. Un análisis comparativo*. Eudema. Madrid (1995).

[14] Jonathan D. Sime: *Design Against Fire: An Introduction to Fire Safety Engineering Design*. E & F.N. Spoon. Londres (1994).

[15] Guylene Proulx, Jonathan D. Sime: *To Prevent 'Panic' in a Underground Emergency: Why Not Tell People The Truth?. Fire Safety Science*. (1991).

[16] Sigmund Freud: *Psicología de las masas y análisis del yo*. International Psycoanalytic Publishing House. Viena (1921).

[17] Stanley Milgram, Hans Toch: *Collective Behavior: Crowds and Social Movements*. *The Handbook of Social Psychology*. Addison-Wesley. USA (1969).

[18] Ralph Turner, Lewis M. Killian: *Collective Behavior 2d*. Prentice Hall. USA (1972).

[19] Enrique E. Tarifa: *Teoría de Modelos y Simulación*. Facultad de Ingeniería. Universidad Nacional de Jujuy. Argentina.

[20] Fernando Sancho: *Sistemas Multiagente y Simulación*. Sevilla (2016). [Consulta: 18 jun. 2017]. Disponible en <http://www.cs.us.es/~fsancho/?e=57>

[21] Colaboradores de Wikipedia: *Sistemas multiagente*. Wikipedia (2017). [Consulta: 20 jun. 2017]. Disponible en <https://es.wikipedia.org/wiki/Sistema_multiagente>

[22] Michael J. Wooldridge, Nicholas R. Jennings: *Agent Theories, Architectures, and Languages: a Survey*. Intelligent Agents, Wooldridge and Jennings Eds. Springer-Verlag. United Kingdom (1995).

[23] Jorge J. Gómez: *Metodologías para el desarrollo de sistemas multi-agente*. Facultad de Informática. Universidad Complutense de Madrid (2003).

[24] Jorge J.Gómez, Carlos R. Fernández, Javier Arroyo: *Model Driven Development and Simulations with the INGENIAS Agent Framework*. Facultad de Informática. Universidad Complutense de Madrid (2010).

[25] Colaboradores de Wikipedia: *Ecuaciones Lotka-Volterra*. Wikipedia (2017). [Consulta: 23 jun. 2017]. Disponible en <https://es.wikipedia.org/wiki/Ecuaciones_Lotka-Volterra>

[26] Nuria Badenes Plá: *Microsimulación y Economía de la salud*. Cuadernos económicos de ICE, n.º 75. Ministerio de Economía, Industria y Competitividad. Gobierno de España (2008).

[27] Craig W. Reynolds: *Flocks, Herds and Schools. A Distributed Behavioral Model*. Computer Graphics volume 21, number 4. USA (1987).

[28] Alec Banks, Jonathan Vincent, Chukwudi Anyakoha: *A review of particle swarm optimization. Part I: background and development*. Natural Computing. UK (2007).

[29] Colaboradores de Wikipedia: *Boids*. Wikipedia (2017). [Consulta: 26 jun. 2017]. Disponible en <https://en.wikipedia.org/wiki/Boids>

[30] Iztok Lebar, Frank H. Heppner: *Organizated flight in birds*. Animal Behaviour Journal (2009).

[31] Jose L. Ferreira: *Recordando a Schelling y su modelo de segregación*. [Consulta: 28 jun. 2017]. Disponible en <http://nadaesgratis.es/jose-luis-ferreira/recordando-a-schelling-y-su-modelo-de-segregacion>

[32] Thomas C. Schelling: *Micromotives and macrobehavior*. WW Norton & Company. USA (2006).

[33] Vi Hart, Nicky Case, David T. Marchand: *Parábola de los polígonos*. [Consulta: 1 jul. 2017]. Disponible en <http://ncase.me/polygons-es/>

[34] Joshua M. Epstein, Robert Axtell: *Growing artificial societies: social science from the bottom up*. Brooking Institute Press. The MIT Press. USA (1996).

[35] *The Ascape Model Developer's Manual*. Sourceforge. (2010).

[36] *Sugarscape: Agent-Based Modeling – Wolfram Demonstrations Project*. Wolfram. (2011).

[37] Anthony Bigbee, Claudio Cioffi-Revilla, Sean Luke: *Replication of Sugarscape Using MASON. Agent-Based Approaches in Economic and Social Complex Systems IV: Post-Proceedings of The AESCS International Workshop 2005*. Springer. Tokyo (2005).

[38] David Canter, John Breaux, Jonathan D. Sime: *Domestic, multiple occupancy and Hospital fires*. In D.Canter (ed.) Fires and human behaviour. USA (1980).

[39] Juan I. Aragonés, Fernando Talayero: *La conducta humana en los incendios*. Papeles del psicólogo, N.º 68 (1997).

[40] Robert L. Paulsen: *Human behavior and fires: An introduction.* Fire Technology volume 20, number 2 (1984).

[41] Juan I. Aragonés, Fernando Talayero: *Un análisis de contenido de las informaciones sobre los incendios urbanos en la prensa*. Universidad de Barcelona (1996).

[42] Rafael Pax: *What is MASSIS?*. [Consulta: 3 jul. 2017]. Disponible en <http://www.massisframework.com/>

[43] Manuel Fidalgo: *La conducta humana ante situaciones de emergencia: análisis de proceso en la conducta individual*. Instituto Nacional de Seguridad e Higiene en el Trabajo. Barcelona (1995).

[44] Manuel Fidalgo: *La conducta humana ante situaciones de emergencia: la conducta colectiva*. Instituto Nacional de Seguridad e Higiene en el Trabajo. Barcelona (1995).

[45] Jason Van Zyl: *History of Maven*. [Consulta: 6 jul. 2017]. Disponible en <https://maven.apache.org/background/history-of-maven.html>

[46] Página web de centro comercial: Gran Vía de Hortaleza. [Consulta: 9 jul. 2017]. Disponible en <http://es.club-onlyou.com/Gran-Via-Hortaleza/Horario-y-Accesos#ancreCoordonnees>

[49] Yves Demazeau: *From cognitive interactions to collective behaviour in agent-based systems*. European Conference of Cognitive Science. France (1995).

[50] Carlos A. Iglesias: *Definición de una metodología para el desarrollo de Sistemas Multi-Agente*. Tesis doctoral. Departamento de ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid (1998).

[51] Carlos A. Iglesias, Mercedes Garijo, José C. González: *Metodologías orientadas a agentes. Inteligencia Artificial*. Revista Iberoamericana de Inteligencia Artificial. Número 6, Volumen 2 (1998).

[52] David Kinny, Michael Georgeff, Anand Rao: *A Methodology and Modelling Technique for Systems of BDI Agents*. MAAMAW (1996).

[53] Scott A. DeLoach: *Analysis and Design using MaSE and agentTool*. Proceedings of the 12th Midwest Artifical Intelligence and Cognitive Science Conference (MAICS) (2001).

[54] Michael Wooldridge, Nick Jennings, Davis Kinny: *The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and multi-agent systems*. Springer (2000).

[55] Jorge J. Gómez: *Modelado de Sistemas Multi-agente*. Tesis, Facultad de Informática, Universidad Complutense de Madrid (2002).

[56] Jorge J. Gómez, Rubén Fuentes: *The INGENIAS Methodology*. Fourth Iberoamerican Workshop on Multi-agent Systems Iberagents 2002.

[57] Guy H. Orcutt: *A new type of socioeconomic system. Review of Economics and Statics*. MIT Press. USA (1957).